Российская академия наук
Отделение математических наук РАН
Вычислительный центр им. А. А. Дородницына РАН

Национальная академия наук Украины
Крымская академия наук
Таврический национальный университет им. В. И. Вернадского

при поддержке

Российского фонда фундаментальных исследований
Компании FORECSYS

Интеллектуализация обработки информации (ИОИ-8)

Кипр, г. Пафос,

17–23 октября 2010

Избранные доклады (Воронцов и K°)

Содержание

Точные комбинаторные оценки обобщающей способности онлайнового обучения*

Воронцов К. В.[1]***, Решетняк И. М.***[2]
vokov@forecsys.ru[1], ilya.reshetnyak@gmail.com[2]
Москва, [1]ВЦ РАН, [2]ВМК МГУ

Рассматриваются задачи онлайнового обучения, когда обучающие объекты поступают последовательно, и в каждый момент времени выбирается лучший предиктор по критерию минимума числа ошибок на предыстории. Для нескольких модельных семейств предикторов получены точные формулы, выражающие зависимость вероятности ошибки от времени. Показано, что обучение гораздо эффективнее в тех случаях, когда семейство предикторов обладает свойствами расслоения и связности.

Exact combinatorial generalization bounds for online learning*

Vorontsov K. V.[1]***, Reshetnyak I. M.***[2]
Moscow, [1]Computing Center RAS, [2]Moscow State University

The online learning framework is considered, in which training objects come subsequently, and the better predictor is chosen at each moment from the empirical risk minimization. Exact combinatorial formulae are obtained that describe the probability of error as a function of time for several artificial sets of predictors. The learning is shown to be much more effective when the set of predictors possesses the properties of splitting and similarity.

В теории статистического обучения существует много подходов к оцениванию обобщающей способности алгоритмов классификации [3]. Многие оценки, начиная с классических [2], служат обоснованием известного эмпирического факта, что выбор алгоритма по критерию минимума частоты ошибок на обучающей выборке может приводить к *переобучению*, т. е. к существенно более высокой частоте ошибок на контрольной выборке. Как правило, переобучение тем сильнее, чем сложнее семейство алгоритмов, т. е. чем больше в нём *различимых* алгоритмов (два алгоритма *неразличимы*, если они допускают ошибки на одних и тех же объектах обучающей и контрольной выборки). Эксперименты [6] и точные комбинаторные оценки [7] показали, что переобучение зависит не только от числа различных алгоритмов, но и от степени их различности. Одновременное наличие свойств расслоения и связности у семейства алгоритмов приводит к радикальному снижению переобучения.

Будем говорить, что алгоритмы, допускающие m ошибок на объединённой выборке, образуют m-й слой. *Эффект расслоения* связан с тем, что в нижних слоях, как правило, находится ничтожно малая доля алгоритмов, однако чем ниже слой, тем выше вероятность получить алгоритм из этого слоя в результате обучения. Это приводит к сокращению «эффективной сложности» семейства, так как фактически используется лишь небольшая его часть.

Эффект связности возникает в семействах алгоритмов, непрерывных по параметрам. В этом случае для каждого алгоритма в семействе найдётся некоторое число алгоритмов, различимых с ним только на одном каком-то объекте полной выборки. Будем называть такие алгоритмы связанными. Схожие (в частности, связанные) алгоритмы вносят меньший вклад в вероятность переобучения, чем несхожие.

Показано [6], что семейства, состоящие всего из нескольких десятков алгоритмов и не обладающие свойствами расслоения или связности, могут иметь столь высокую вероятность переобучения, что их применение окажется нецелесообразным. В то же время, расслоенные и связные семейства могут содержать миллиарды алгоритмов, обладая при этом незначительным переобучением. Таким образом, сочетание этих двух эффектов является критически важным.

Точные оценки вероятности переобучения получены к настоящему моменту для ряда модельных семейств алгоритмов [1, 7, 4]. Эти семейства обладают некоторой «искусственной» структурой, что и позволяет выводить для них точные оценки с помощью комбинаторных методов. Такие семейства вряд ли могут порождаться практическими задачами. С другой стороны, они обладают теми же ключевыми свойствами, что и реальные семейства — расслоением, связностью, размерностью. Показано [1], что вероятность переобучения нейронных сетей, решающих деревьев, байесовских классификаторов на реальных задачах классификации неплохо приближается вероятностью переобучения модельного семейства — многомерной сетки алгоритмов, при соответствующем подборе её размерности. Вообще, изучение модельных семейств представляется необходимым промежуточным этапом построения комбинаторной теории переобучения.

Работа поддержана РФФИ (проект №08-07-00422) и программой ОМН РАН «Алгебраические и комбинаторные методы математической кибернетики и информационные системы нового поколения».

В данной работе исследуются задачи динамического или *онлайнового* (on-line) обучения [8], в которых обучающие объекты поступают потоком, и при получении каждого нового объекта необходимо корректировать текущий алгоритм (алгоритмы в онлайновых задачах будем называть *предикторами*). Рассматривается частная постановка задачи онлайнового обучения, для которой основные конструкции комбинаторной теории переобучения [7] переносятся практически без изменений. Методом порождающих и запрещающих множеств выводится точное выражение для *кривой обучения* — зависимости вероятности ошибки на следующем объекте от времени. Аналогично [6] вводится модельное семейство с расслоением и связностью — *монотонная цепочка* предикторов, а также соответствующие ей семейства, не обладающие свойствами расслоения и/или связности. Оказывается, что в случае онлайнового обучения вероятность ошибки также существенно понижается при одновременном наличии свойств расслоения и связности.

Задача онлайнового обучения

Общая постановка задачи. Имеется конечная последовательность объектов $X^T = (x_1, \ldots, x_T)$ и множество допустимых предсказаний A. Природа множеств X^T и A не оговаривается. В каждый момент времени t *предсказывающий алгоритм* μ получает подпоследовательность $X^t = (x_1, \ldots, x_t)$ и выдаёт некоторый элемент множества A — предсказание $a_{t+1} = \mu(X^t)$ относительно момента времени $t{+}1$. После того, как становится известен объект x_{t+1}, вычисляется величина потерь, возникших вследствие неточности сделанного предсказания $\mathscr{L}(a_{t+1}, x_{t+1})$, где $\mathscr{L}\colon A \times X^T \to \mathbb{R}$ — заданная *функция потерь*. Задача заключается в том, чтобы найти *кривую обучения* — зависимость математического ожидания потери от времени

$$Q(t) = \mathsf{E}\mathscr{L}(a_{t+1}, x_{t+1}), \quad t = 1, \ldots, T-1.$$

Темп её убывания характеризует обучаемость (способность к обобщению эмпирических фактов) предсказывающего алгоритма μ.

Вероятностная модель данных. Математическое ожидание будем понимать в смысле слабой вероятностной аксиоматики [5, 6, 7], полагая, что множество объектов X^T не случайно, но случаен порядок их появления, причём все $T!$ перестановок могут реализоваться с равной вероятностью. Это аналог стандартного предположения о том, что объекты выбираются случайно и независимо из неизвестного, но фиксированного распределения на множестве всех допустимых объектов, вообще говоря, бесконечном. Мы пользуемся ослабленным вероятностным предположением, не требующим введения бесконечных множеств и σ-алгебр на бесконечных множествах. При данном предположении случайная величина ξ — это произвольная функция $\xi(X^T)$ последовательности X^T, а математическое ожидание определяется как среднее по всем перестановкам, $\mathsf{E}\xi = \frac{1}{T!}\sum_{\sigma \in S_T} \xi(\sigma X^T)$, где S_T — симметрическая группа, σX^T — последовательность, полученная в результате действия перестановки σ на исходной последовательности X^T.

Частная постановка задачи. Сделаем три дополнительных предположения.

1. Пусть $A = \{a\colon X \to R\}$ — множество функций от объекта, называемых *предикторами*.

2. Пусть функция потерь бинарна: $\mathscr{L}(a, x) = 1$, если предиктор $a \in A$ ошибается на объекте $x \in X^T$, и $\mathscr{L}(a, x) = 0$, если a не ошибается на x.

Обозначим через $n(a, X)$ число ошибок предиктора a на множестве объектов X, а через $\nu(a, X) = \frac{1}{|X|} n(a, X)$ — частоту ошибок a на X.

3. Пусть алгоритм μ выбирает предиктор с минимальным числом ошибок на предыстории:

$$a_{t+1} = \mu(X^t) = \arg\min_{a \in A} n(a, X^t).$$

Природа множества R не оговаривается. В частности, если R — конечное множество классов, то естественно полагать $\mathscr{L}(a, x) = \big[a(x) \neq y^*(x)\big]$, где $y^*(x)$ — правильная классификация объекта x. Если $R = \mathbb{R}$, то имеем регрессионную задачу, в которой можно полагать $\mathscr{L}(a, x) = \big[|a(x) - y^*(x)| > \delta\big]$, где δ — заданный порог ошибки.

Бинарный вектор $\bar{a} = \big(\mathscr{L}(a, x_t)\big)_{t=1}^T$ будем называть *вектором ошибок* предиктора a. Векторы ошибок $\bar{a}$ всех предикторов $a \in A$ образуют *матрицу ошибок* размера $T \times |A|$. Будем полагать, что все столбцы этой матрицы попарно различны.

Если каждые два соседних столбца отличаются только по одной строке (хэммингово расстояние между столбцами равно единице), то множество A будем называть *цепочкой* предикторов.

Заметим, что $\mu(X^t)$ не зависит от порядка объектов в подпоследовательности X^t.

Кривые обучения для некоторых модельных семейств предикторов

Лемма 1. *Если значение случайной величины* $\xi(X^T) \equiv \xi(x_1, \ldots, x_t, x_{t+1})$ *не зависит от объектов* $x_{t+2}, \ldots, x_T$ *и от порядка объектов* $x_1, \ldots, x_t$, *то*

$$\mathsf{E}\xi = \frac{1}{C_T^t (T-t)} \sum_{X^t \subseteq X} \sum_{x \in X \setminus X^t} \xi(X^t, x),$$

где X^t — *всевозможные неупорядоченные подмножества мощности* t *множества объектов* X.

Функция потерь $\mathscr{L}(\mu(X^t), x_{t+1})$ удовлетворяет условию Леммы 1, откуда вытекает

Лемма 2. *Кривая обучения $Q(t)$ совпадает с зависимостью функционала полного скользящего контроля от длины обучающей выборки t:*

$$Q(t) = \frac{1}{C_T^t} \sum_{X \sqcup \bar{X}} \nu(\mu(X), \bar{X}),$$

где $X \sqcup \bar{X} = X^T$ — всевозможные разбиения множества X^T на два неупорядоченных подмножества — обучающую выборку X длины t и контрольную выборку $\bar{X}$ длины $T-t$.

Семейства из одного и двух элементов для обычного (off-line) обучения рассмотрены в [6, 7]. Рассмотрим их в случае онлайнового обучения.

Теорема 3. *Для семейства $A = \{a\}$, состоящего из одного предиктора с известным $n(a, X^T) = m$, кривая обучения не зависит от времени: $Q(t) = \frac{m}{T}$.*

Теорема 4. *Пусть $A = \{a_0, a_1\}$ и в X^T имеется ровно m_0 объектов, на которых ошибается только a_0, ровно m_1 объектов, на которых ошибается только a_1 и ровно m_2 объектов, на которых ошибаются оба предиктора. Тогда*

$$Q(t) = \sum_{s_0=0}^{m_0} \sum_{s_1=0}^{m_1} \sum_{s_2=0}^{m_2} \frac{C_{m_0}^{s_0} C_{m_1}^{s_1} C_{m_2}^{s_2} C_{T-m_0-m_1-m_2}^{t-s_0-s_1-s_2}}{C_T^t (T-t)} \times$$
$$\times \big((m_0 - s_0)[s_0 < s_1] + (m_1 - s_1)[s_0 \geqslant s_1] + m_2 - s_2\big).$$

Исследование поведения кривой $Q(t)$ в зависимости от разности слоёв $|m_0 - m_1|$ и от хэммингова расстояния между предикторами $m_0 + m_1$ показывает, что эффекты расслоения и сходства проявляются даже в этом простейшем случае, когда предикторов только два, и способствуют уменьшению вероятности ошибки, рис. 1.

Метод порождающих и запрещающих множеств [7] также переносится на случай онлайнового обучения.

Гипотеза 1. При любом $t = 1, \ldots, T-1$ для каждого предиктора $a \in A$ можно указать подмножества $X_a, X'_a \subset X^T$ такие, что для всех t-элементных подмножеств $X \subset X^T$

$$\big[\mu(X){=}a\big] = \big[X_a \subseteq X\big]\big[X'_a \subseteq \bar{X}\big].$$

Множество X_a называется *порождающим*, X'_a — *запрещающим* для предиктора a.

Теорема 5. *Если справедлива Гипотеза 1, то*

$$Q(t) = \sum_{a \in A} \frac{P(a)}{T-t} \left(\frac{t_a}{T_a} n(a, X'_a) + \frac{T_a - t_a}{T_a} n(a, X^T) \right),$$

где $P(a) = \mathsf{E}\big[\mu(X){=}a\big] = C_{T_a}^{t_a} / C_T^t$ — вероятность получить предиктор a в результате обучения, $T_a = T - |X_a| - |X'_a|$, $t_a = t - |X_a|$.

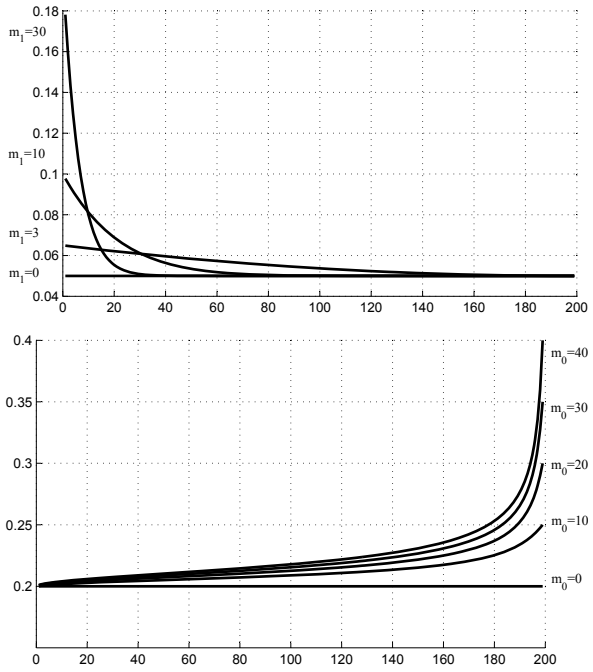

Рис. 1. Кривая обучения $Q(t)$ для пары предикторов, $T = 200$. Вверху: $m_0 = 0$, $m_1 = 0, 3, 10, 30$, $m_2 = 10$. Внизу: $m_0 = m_1 = 0, 10, 20, 30, 40$, $m_2 = 40 - m_0$.

Теорема 5 позволяет эффективно вычислять $Q(t)$ для некоторых модельных семейств предикторов, в частности, для монотонной цепочки.

Монотонная цепочка [7] — это множество предикторов $A = \{a_0, \ldots, a_D\}$, удовлетворяющее двум требованиям:

1) $n(a_d, X^T) = m + d$ для всех $d = 0, \ldots, D$;
2) $\mathscr{L}(a_{d-1}, x) \leqslant \mathscr{L}(a_d, x)$ для всех $d = 1, \ldots, D$ и всех $x \in X^T$.

Монотонная цепочка является идеализированной моделью связного семейства предикторов с одним параметром, предполагающая, что при непрерывном отклонении параметра в сторону от оптимального значения число ошибок предиктора на полной выборке монотонно не убывает.

Теорема 6. *Для монотонной цепочки $\{a_0, \ldots, a_D\}$*

$$Q(t) = \sum_{d=0}^{\min(D, T-t)} \frac{C_{T-d-1}^{t-1}}{C_T^t (T-t)} R_T^t(d, m),$$

$$R_T^t(d, m) = \begin{cases} d + m\big(\frac{T-d-t}{T-d-1}\big), & d < D; \\ d\big(\frac{T-d}{t}\big) + m\big(\frac{T-d-t}{t}\big), & d = D. \end{cases}$$

На рис. 2 показаны кривые $Q(t)$ для различных значений числа D предикторов в цепочке. Начиная с $t \sim 40$ вероятность ошибки практически полностью определяется несколькими лучшими предикторами, т. е. нижними слоями семейства. При этом вероятность ошибки не сильно отличается от $\frac{m}{T}$ — частоты ошибок лучшего предиктора.

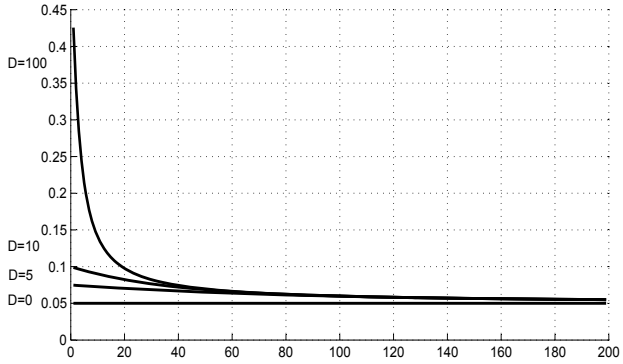

Рис. 2. Кривая обучения $Q(t)$ для монотонной цепочки предикторов, при $T = 200,\ m = 10,\ D = 0, 5, 10, 100.$

Вычислительный эксперимент

Для случая оффлайнового обучения в [6] описан эксперимент, целью которого было определить, какое из двух свойств — расслоение или связность — более важно для уменьшения переобучения. Идея эксперимента заключалась в том, чтобы сравнить четыре семейства: *цепочку с расслоением*, в которой лучший алгоритм допускает m ошибок; *цепочку без расслоения*, в которой все алгоритмы допускают m или $m+1$ ошибок; и ещё две *не-цепочки*, которые получаются из первых двух семейств путём случайной перестановки элементов в каждом столбце матрицы ошибок. Основной вывод [6] заключался в том, что важно наличие обоих свойств одновременно, а семейства, не обладающие хотя бы одним из двух свойств, могут сильно переобучаться уже при $|A|$ порядка нескольких десятков. Поскольку для не-цепочек удобных теоретических оценок пока не найдено, эксперимент проводился методом Монте-Карло по 10^4 случайных разбиений на обучающую и контрольную выборки.

В данной работе аналогичный эксперимент ставится для задачи онлайнового обучения. Матрицы ошибок четырёх семейств предикторов строятся точно так же. Методом Монте-Карло генерируется 10^4 случайных перестановок выборки. Основное отличие этого эксперимента в том, что вместо зависимости вероятности переобучения от $|A|$ строятся и сравниваются кривые обучения $Q(t)$, рис. 3.

Выводы

Принятая нами вероятностная модель данных основана на предположении, что порядок поступления объектов не зависит от времени. Именно это и позволило перенести основные конструкции комбинаторной теории переобучения [5–7] на задачи онлайнового обучения и получить в явном виде зависимости вероятности ошибки от времени. Показано, что, как и в обычных (оффлайновых) задачах обучения, эффекты расслоения и связности в семействах предикторов способствуют уменьшению вероятности ошибки.

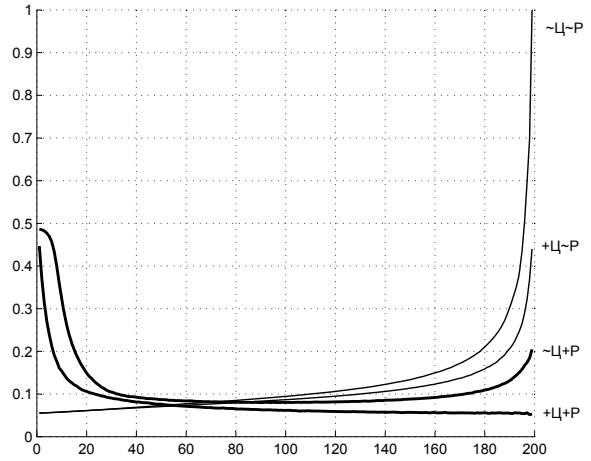

Рис. 3. Кривая обучения $Q(t)$ для четырёх семейств: цепочка с расслоением (+Ц+Р), не-цепочка с расслоением (~Ц+Р), цепочка без расслоения (+Ц~Р), не-цепочка без расслоения (~Ц~Р), при $T = 200,\ |A| = 200,\ m = 10.$

Заметим, что предложенная вероятностная модель неприменима в случае нестационарных процессов обучения, когда вероятность ошибки отдельных предикторов меняется во времени. В дальнейшем это ограничение модели предполагается снять путём введения ограничений на множество допустимых перестановок.

Литература

[1] *Ботов П. В.* Точные оценки вероятности переобучения для монотонных и унимодальных семейств алгоритмов // Всеросс. конф. ММРО-14. — М.: МАКС Пресс, 2009. — С. 7–10.

[2] *Вапник В Н., Червоненкис А. Я.* Теория распознавания образов. — М.: Наука, 1974.

[3] *Boucheron S., Bousquet O., Lugosi G.* Theory of classification: A survey of some recent advances // ESAIM: Probability and Statistics. — 2005. — No. 9. — Pp. 323–375.

[4] *Frey A. I.* Accurate estimates of the generalization ability for symmetric sets of predictors and randomized learning algorithms // Pattern Recognition and Image Analysis. — 2010. — Vol. 20, No. 3. — Pp. 241–250.

[5] *Vorontsov K. V.* Combinatorial probability and the tightness of generalization bounds // Pattern Recognition and Image Analysis. — 2008. — Vol. 18, No. 2. — Pp. 243–259.

[6] *Vorontsov K. V.* Splitting and similarity phenomena in the sets of classifiers and their effect on the probability of overfitting // Pattern Recognition and Image Analysis. — 2009. — Vol. 19, No. 3. — Pp. 412–420.

[7] *Vorontsov K. V.* Exact combinatorial bounds on the probability of overfitting for empirical risk minimization // Pattern Recognition and Image Analysis. — 2010. — Vol. 20, No. 3. — Pp. 269–285.

[8] *Vovk V., Gammerman A., Shafer G.* Algorithmic learning in a random world. — Springer Science+Business Media, Inc., 2005. — 324 p.

Application of monotonic correction to learning to rank

Spirin N. V.
nikita.spirin@phystech.edu
Moscow Institute of Physics and Technology, Dolgoprudny, Russia

Learning to rank is one of the hottest problems in web search. A lot of successful algorithms solving this problem have been proposed over the last decade. One of the most popular approach is an application of boosting techniques to learning to rank. The general idea of boosting is to build a large set of weak rankers and then combine them efficiently in a *linear* composition. Despite the fact that this method is generally successful for applications, it isn't effective when used to boost the performance of individually strong algorithms. To address this problem we propose a novel algorithm, that is able to blend base rankers efficiently, based on a theory of compositions with *nonlinear* monotonic correcting functions. In this paper we also provide an auxiliary algorithm to monotonize a set of vectors — a subproblem which is necessary to solve before building a monotonic composition. Experiments with real data show that our achieves high accuracy and outperforms existing state-of-the-art algorithms.

The learning to rank problem

A retrieval system functions as follows. Given a query and a trained ranking function, an IR system returns a list of documents that are somehow related to the query and sorted in descending order of relevance according to the ranking function. The latter is formed during the learning stage. As a training set we have a set of queries and documents associated with them. In addition to that, for each query-document pair from this set there is a relevance score. Relevance scores form an ordered set and indicate the level of usefulness of a document to a query. According to the general machine learning approach the objective of *learning to rank* (L2R) is to synthesize the ranking function that maximizes a performance measure.

If stated formally, the structure of the training set looks as follows. Given an ordered set of ranks $\mathbb{Y} = \{r_1, \dots, r_K\}$ and a set of queries $Q = \{q_1, \dots, q_n\}$. A list of documents $D_q = \{d_{q1}, \dots, d_{q,n(q)}\}$ is associated with each query $q \in Q$, where $n(q)$ is the number of documents associated with the query q. A factor ranking model is admitted, i.e. each query-document pair (q, d), $d \in D_q$ is represented by the vector of features $\boldsymbol{x}_{qd} = \big(f_1(q,d), \dots, f_m(q,d)\big) \in \mathbb{R}^m$ and the corresponding relevance score $r(q,d) \in \mathbb{Y}$. Finally, the training set can be represented as

$$S = \{\boldsymbol{x}_{qd}, r(q,d)\}, \quad q \in Q,\ d \in D_q.$$

The objective is to build a ranking function $A\colon \mathbb{R}^m \to \mathbb{Y}$ that maximizes a performance measure on the training set and on a new testing dataset as well.

Monotonic correcting functions

Monotonicity restriction is very natural to impose on correcting function that aggregate outputs of weak base predictors [3, 6]. Indeed, if output of a predictor is higher for an object, provided that outputs of other predictors are the same, then the output of the entire composition must be also higher for this object. This implies that the correcting function is to be monotonic. Obviously, linear composition (weighted voting) meets the monotonicity restriction if only all the weights are nonnegative. Monotonicity is a less restrictive principle than the weighted voting is. Moreover, monotonic correcting function is more flexible in case of L2R, and allows to build small compositions of strong rankers more efficiently. In this section we restate the general framework of monotonic compositions in case of classification and regression [3]. Later we will adapt it for learning to rank.

Let Ω be an object space, X and Y be partially ordered sets, B be a set of base algorithms $b\colon \Omega \to X$, X is referred to as an estimation space (predictions of base algorithms) and Y as an output space (labels, responses, relevance scores, etc.).

A training set of object–output pairs $\{(x_k, y_k)\}_{k=1}^{\ell}$ from $X \times Y$ and a set of base algorithms $b_1, \dots, b_p$ induces a sequence of estimation vectors $\{\boldsymbol{u}_k\}_{k=1}^{\ell}$ from X^p, where $\boldsymbol{u}_k = \big(u_k^1 = b_1(x_k), \dots, u_k^p = b_p(x_k)\big)$.

Define an order on X^p: $(u^1, \dots, u^p) \leqslant (v^1, \dots, v^p)$, if $u^i \leqslant v^i$ for all $i = 1, \dots, p$. If $\boldsymbol{u} \neq \boldsymbol{v}$ and $\boldsymbol{u} \leqslant \boldsymbol{v}$, then $\boldsymbol{u} < \boldsymbol{v}$. A map $F\colon X^p \to Y$ is referred to as monotonic, if $\boldsymbol{u} \leqslant \boldsymbol{v}$ implies $F(\boldsymbol{u}) \leqslant F(\boldsymbol{v})$ for all $\boldsymbol{u}, \boldsymbol{v} \in X^p$.

Monotonic composition of base algorithms $b_1, \dots, b_p$ with *monotonic correcting function* F is a map $a\colon \Omega \to Y$ defined as $a(x) = F\big(b_1(x), \dots, b_p(x)\big)$ for all $x \in \Omega$.

If the base algorithms are fixed, then the learning of function F from data can be stated as a task of monotonic interpolation. Given a sequence of vectors $\{\boldsymbol{u}_k\}_{k=1}^{\ell}$ from X^p and a sequence of targets $\{y_k\}_{k=1}^{\ell}$ from Y, one should build such a monotonic function F that meets the *correctness condition*

$$F(\boldsymbol{u}_k) = y_k, \quad k = 1, \dots, \ell. \tag{1}$$

Sequence of pairs $\{(\boldsymbol{u}_k, y_k)\}_{k=1}^{\ell}$ is called monotonic, if from $\boldsymbol{u}_i < \boldsymbol{u}_j$ follows $y_i \leqslant y_j$ for all $i, j = 1, \dots, \ell$. Obviously, a monotonic function F meeting the condition (1) exists **iff** the sequence $\{(\boldsymbol{u}_k, y_k)\}_{k=1}^{\ell}$ is monotonic.

A pair (i, j) is called *defective* for the base algorithm b, if $b(x_i) \geqslant b(x_j)$ and $y_i < y_j$.

Denote a set of all defective pairs of b as $\mathbb{D}(b)$.

Международная конференция «Интеллектуализация обработки информации» (ИОИ-8), Кипр, г. Пафос, 17–23 октября 2010 г.
International conference "Intelligent Information Processing" IIP-8, Cyprus, Paphos, October 17–23, 2010.

A set $\mathbb{D}(b_1,\dots,b_p) = \mathbb{D}(b_1)\cap\dots\cap\mathbb{D}(b_p)$ is called *cumulative defect* of a set of base algorithms $(b_1,\dots,b_p)$.

The number of defective pairs $|\mathbb{D}(b)|$ play a role of a quality measure for a base algorithm b.

In composition, if we build the next base algorithm b_{p+1} so that it yields a right order on pairs (i,j) from cumulative defect of preceding base algorithms,

$$b_{p+1}(x_i) < b_{p+1}(x_j) : \ (i,j) \in \mathbb{D}(b_1,\dots,b_p), \qquad (2)$$

then $\mathbb{D}(b_1,\dots,b_p,b_{p+1}) = \varnothing$, and a monotonic function F meeting (1) will exist.

The more accurate analysis of the difference between two sets $\mathbb{D}(F(b_1,\dots,b_p)) \setminus \mathbb{D}(b_1,\dots,b_p)$ show that not only the cumulative defect but also defective pairs constituting the so called defective triplets impede the correctness condition [3]. Any defective triplet contains a defective pair from cumulative defect, however, a defective pair may induce many defective triplets. Thus, the system (2) acquires weights w_{ij} for each inequality $(i,j) \in \mathbb{D}(b_1,\dots,b_p)$. The task is to find the heaviest consistent sub-system of inequalities. Below we restate this problem in terms of quality functional and analyze its properties for L2R problem.

The MonoRank algorithm

Inspired by outstanding performance of monotonic compositions on classification and regression problems [3, 6], we applied the notion of monotonic correction to L2R problem and developed a novel algorithm for it. The algorithm is referred to as MonoRank and the pseudocode is presented in Algorithm 1.

In order to adapt the optimization task stated in [3] to L2R we substitute query-document pairs (q,d) and (q,d') for objects x_i, x_j respectively, and then restate the problem in question as a minimization of quality functional $\mathscr{Q}$ with base algorithms $b_1,\dots,b_p$ fixed:

$$\mathscr{Q}(b_1,\dots,b_p,b_{p+1}) = \sum_{q\in Q}\sum_{(d,d')} w_{qdd'}\big[b_{p+1}(q,d) \geqslant b_{p+1}(q,d')\big] \to \min_{b_{p+1}}, \qquad (3)$$

where (d,d') are all documents from the list D_q such that $(d,d') \in \mathbb{D}(b_1,\dots,b_p)$.

In classification and regression tasks special efforts are to be made to reduce the pairwise criterion $\mathscr{Q}$ to usual pointwise empirical risk [3]. In learning to rank such tricks are needless as long as base rankers can be learned directly from $\mathscr{Q}$ minimization, particularly in pairwise approach.

Base rankers for monotonic correction

Having analyzed successful L2R algorithms, in this section we present their modifications to be used in monotonic compositions. The general idea is to use weights $w_{qdd'}$ for pairs of documents.

The weighted-RankSVM algorithm. Insertion of weights in RankSVM algorithm doesn't face any difficulties [2]. The only thing to do is to add weights $w_{qdd'}$ whenever we come across slack variables in the objective function. In this case the classical SVM quadratic programming task will look like

$$\min_{\alpha,\,\{\xi_{qdd'}\}} \|\boldsymbol{\alpha}\|^2 + C\sum_{q\in Q}\sum_{d,d'\in D_q} w_{qdd'}\xi_{qdd'};$$
$$\boldsymbol{\alpha}^\intercal(\boldsymbol{x}_{qd} - \boldsymbol{x}_{qd'}) \geqslant 1 - \xi_{qdd'},$$
$$\xi_{qdd'} \geqslant 0, \quad q\in Q;\ (d,d') \in D_q;.$$

To sum up, we append an additional vector of weights $\{w_{qdd'}\}$ as a parameter of the algorithm.

The weighted-RankBoost algorithm. RBoost already contains the distribution of weights $w_{qdd'}$ over the set of document pairs $q \in Q,\ d,d' \in D_q$. Usually the weights are initialized by the uniform distribution, and updated at each subsequent iteration t so that the weak ranker b_t focus more on the document pairs that was ranked incorrectly by previous weak rankers. See the original text for additional information [1]. In order to use RankBoost in monotonic composition we propose to initialize RankBoost by normalized weights $w_{qdd'}$ from (3) instead of uniform distribution.

Other L2R algorithms from pairwise family can be modified analogously to be used as base rankers for monotonic composition. For example, weights $w_{qdd'}$ can be easily inserted in FRank [4] and RankNet [7].

Algorithms from pointwise family, e. g. McRank [8], can be modified by introducing document weights $w_{qd} = \sum_{d'\in D_q} w_{qdd'} + w_{qd'd}$.

Reweighing strategies

Now let us discuss the initialization of weights $w_{qdd'}$ for algorithms described above in order for them to form a strong and diversified set of base rankers. Particularly, below we will describe various reweighing strategies.

Algorithm 1. MonoRank pseudocode.

Input: training set $S = \{\boldsymbol{x}_{qd}, r(q,d)\}, q\in Q, d\in D_q$;
δ — number of unsuccessful iterations before stop;
Output: ranking composition $M_T(q,d)$ of size T;

```
initialize weights w_{qdd'} = 1,  q ∈ Q,  d, d' ∈ D_q;
for t = 1, ..., n
    train base ranker b_t(q,d) using weights {w_{qdd'}};
    get predictions b_t(q,d) of b_t on a training set S;
    monotonize {(b_1(q,d), ..., b_t(q,d)), r(q,d)};
    build a composition M_t(q,d) of size t;
    T = arg min_{p: p⩽t} 𝒬(M_p);
    update {w_{qdd'}} using M_t;
    if t − T ⩾ δ then
        return M_T;
```

The defective pairs minimization is the most natural strategy directly induced from the general theory of monotonic correcting functions:

$$w_{qdd'} = \begin{cases} 1, & (qd, qd') \in \mathbb{D}(b_1, \ldots, b_p); \\ 0, & \text{otherwise}, \end{cases}$$

where $d, d' \in D_q$. However, this approach has a significant shortcoming. If we train our base algorithms only using defective pairs the generalization ability of the entire algorithm will diminish and hence it won't be practically applicable.

The defective triplets minimization is almost the same strategy as the previous one and hence inherits all the shortcomings of it. The only difference is that weights might vary from pair to pair and are formed based on a notion of defective triplet [3]:

$$w_{qdd'} = \begin{cases} t^0_{qdd'} + \frac{1}{2} t_{qdd'} + 1, & (qd, qd') \in \mathbb{D}(b_1, \ldots, b_p); \\ 0, & \text{otherwise}, \end{cases}$$

where $t^0_{qdd'}$ is a number of strictly defective triplets containing the defective pair (qd, qd') and $t_{qdd'}$ is the corresponding number of defective triplets. Theoretically, in this case the weight $w_{qdd'}$ can be interpreted as as a more accurate estimate of the number of defective pairs from $\mathbb{D}(F(b_1, \ldots, b_p))$ that will be deleted if we turn out the defect on the pair (qd, qd'). This seems to be a sensible strategy. However, experiments show that it doesn't allow to gain any quality benefit and only increases the complexity of the algorithm. Our conclusion is in agreement with the related research for classification [6].

The defective and clean pairs minimization is the most successful strategy according to our experiments. It combines *the defect minimization principle* and *the struggle for generalization ability* for the entire algorithm [6].

Definition 1. *A pair (i, j) is clean for the base algorithm b, if $b(x_i) < b(x_j)$ and $y_i < y_j$. A set of all clean pairs of b will be denoted as $\mathbb{C}(b)$. A set of all clean pairs of a set of algorithms $(b_1, \ldots, b_p)$ equals $\mathbb{C}(b_1) \cap \cdots \cap \mathbb{C}(b_p)$ and will be denoted as $\mathbb{C}(b_1, \ldots, b_p)$.*

Note that in L2R framework the indexes i, j become qd, qd' respectively. Then the strategy will look like:

$$w_{qdd'} = \begin{cases} w_{\mathbb{D}}(t), & (qd, qd') \in \mathbb{D}(b_1, \ldots, b_p); \\ 1, & (qd, qd') \in \mathbb{C}(b_1, \ldots, b_p); \\ 0, & \text{otherwise}, \end{cases}$$

In this case the weight for a clean pair equals one. And the weight $w_{\mathbb{D}}(t)$ for a defective pair may depend on iteration $t = 1, \ldots, T$. Particularly, it may increase from iteration to iteration in order to lead the training algorithm to turn out the defect on these pairs. We tasted three variants of reweighing strategies in our experiments:

$$w_{\mathbb{D}}(t) = 1 \;\text{ or }\; (t-1) \;\text{ or }\; 2^{t-1}.$$

The strategy of increasing weights $w_{\mathbb{D}}(t)$ can be used to filter outliers in data. Specifically, if a pair continues to be defective for a considerable number of iterations and hence has a big weight we might remove this pair from our dataset and retrain the algorithm.

Monotonic correcting function

In this section we present a novel approach to blending base rankers $b_1, \ldots, b_p$ with nonlinear monotonic correcting function $F(b_1, \ldots, b_p)$. We will build our algorithm so as to minimize the quality functional induced by the cumulative defect $|\mathbb{D}(b_1, \ldots, b_p)|$. At first, it is worth noting that due to the structure of the training set comparable documents d, d' are only those which are associated with the same query, $d, d' \in D_q$, $q \in Q$. Another distinction from the classification case is that we don't really need to meet the correctness condition [3]. The only constraint to meet is to keep the right ordering of documents on the training set.

Provided that documents associated with different queries aren't comparable at all, the quality functional can be rewritten as the sum of functionals, counting defect only for a particular query. We will denote the defect of the entire algorithm with the correcting function F on the query q as $\mathscr{Q}_q(F)$. Then the optimal correcting function F must be a solution of minimization problem

$$\sum_{q \in Q} \mathscr{Q}_q(F) \to \min_F.$$

To give an approximate but computationally efficient solution of this hard problem we propose an averaging heuristic. We solve $|Q|$ problems separately

$$F_q = \arg\min_F \mathscr{Q}_q(F), \quad q \in Q.$$

and give a correcting function F by averaging all F_q:

$$F = \frac{\sum_{q \in Q} |D_q| F'_q}{\sum_{q \in Q} |D_q|}. \tag{4}$$

where F'_q is a normalized to $[0, 1]$ function F_q. Obviously, the function F is monotonic being the sum of monotonic functions.

For illustrative purposes we provide a picture of a monotonic correcting function for ranking, built in accordance with the theory described above and the averaging heuristic. Due to the large number of queries in a training set and hence due to averaging, the monotonic function from the fig. 1 looks like a plane. However, having changed the scale one can observe a complicated texture of the surface. According to our experimental analysis the increase in quality takes place directly thanks to this tiny asperities.

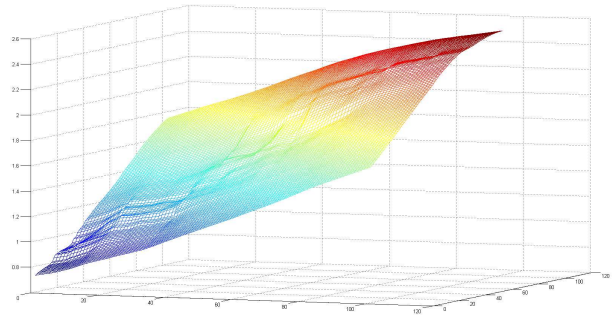

Fig. 1. Monotonic correcting function for ranking.

Experimental results

In this work we use two popular evaluation measures. The first one is the *Mean Average Precision (MAP)* and the second one is the *Discounted Cumulative Gain (DCG)*. RankSVM and RankBoost were used as baseline algorithms. We set the performance of base rankers (RankBoost or RankSVM) with the best values over the ensemble. The MonoRank was chosen from the set of all monotonic compositions of different sizes (number of base algorithms). The choice was guided by the maximum performance on the test set. We utilized a Yahoo! LETOR 2010 dataset [9]. This dataset contains 1266 unique queries, 34815 query-document pairs, 575 features of query-document pairs precomputed and normalized to $[0, 1]$, as well as relevance evaluation made by Yahoo! assessors. Relevance grades are in discrete range $[0, 4]$. We used 5-fold CV to calculate the performance of algorithms. As a base algorithm for MonoRank we used RankBoost due to its computational cost and quality. The results are reported in table 1.

Having seen the fig. 1 one might think why we should use so complicated correcting function to aggregate base rankers. Why couldn't we use just a simple or weighted voting of base rankers? Of course, these are reasonable speculations but we have set an experiment to test the hypothesis. We approximated our "wavy" surface with hyperplane by the least squares method and tested the algorithm on an independent testing set. The results are in table 2.

Conclusion

In this paper a new algorithm called MonoRank is proposed for the learning to rank problem. MonoRank is based on the theory of nonlinear monotonic correcting functions. In distinction from existing methods this algorithm can work as a universal wrapper, able to combine strong algorithms solving the L2R problem. Experimentally, MonoRank algorithm shows high accuracy in ranking and outperforms existing state-of-the-art algorithms, like RankBoost and RankSVM.

Table 1. Results summary.

Metric	MonoRank	RankBoost	RankSVM
MAP	**0.8941**	0.8888	0.8812
NDCG@1	**0.8149**	0.8029	0.8057
NDCG@3	**0.7783**	0.7424	0.7711
NDCG@5	**0.7754**	0.7692	0.7678
NDCG@10	**0.7973**	0.7935	0.7949

Table 2. Comparison of "wavy" monotonic correcting function with its linear approximation.

Metric	MonoRank	MonoRank-linearized
MAP	0.8941	**0.8968**
NDCG@1	**0.8149**	0.8106
NDCG@3	**0.7783**	0.7735
NDCG@5	**0.7754**	0.7720
NDCG@10	**0.7973**	0.7898

References

[1] *Freund Y., Iyer R. D., Schapire R. E., Singer Y.* An efficient boosting algorithm for combining preferences // Journal of Machine Learning Research. — 2003. — Vol. 4, — Pp. 933–969.

[2] *Joachims T.* Optimizing search engines using clickthrough data // SIGIR'08 Conference Proceedings, 133-142, 2002.

[3] *Vorontsov K. V.* Optimization methods for linear and monotone correction in the algebraic approach to the recognition problem // Computational Mathematics and Mathematical Physics. — 2000. — Vol. 40. — Pp. 159–168.

[4] *Tsai M. F., Liu T. Y., Qin T., Chen H. H., Ma W. Y.* FRank: A Ranking Method with Fidelity Loss // SIGIR'07 Conference Proceedings, July 23–27, 2007, Amsterdam.

[5] *R. Schapire* The boosting approach to machine learning: An overview // MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA, 2001.

[6] *Guz I. S.* Nonlinear monotonic compositions of classifiers // Proc. of 13-th Russian conference Mathematical Methods of Pattern Recognition (MMRO-13), Zelenogorsk, 2007.

[7] *Burges C., Shaked T., Renshaw E., Lazier A., Deeds M., Hamilton N., Hullender G.* Learning to Rank using Gradient Descent // In Proceedings of the 22nd International Conference on Machine Learning, 2005.

[8] *Li P., Burges C. J. C., Wu Q.* Learning to Rank Using Classification and Gradient Boosting // Proceedings of the 21st NIPS Conference, Vancouver, Canada, 2007.

[9] Yahoo Learning to Rank Challenge 2010. `http://learningtorankchallenge.yahoo.com/index.php`

Инкрементные методы в коллаборативной фильтрации*

Полежаева Е. А.
lena_polejaeva@mail.ru
Москва, Московский государственный университет им. М. В. Ломоносова, факультет ВМК

В современных приложениях методов коллаборативной фильтрации исходные данные могут иметь большие объёмы (миллионы клиентов и объектов) и поступать в реальном масштабе времени. В этом случае важно требование инкрементности: метод должен эффективно пересчитывать оценки сходства как при появлении новых клиентов и объектов, так и при обновлении значений в матрице исходных данных. Предлагается метод, который не требует хранения матрицы исходных данных и объединяет в себе оба типа инкрементности, используя инкрементное сингулярное разложение (ISVD) и инкрементное вычисление корреляции Пирсона (IPC).

Incremental Methods in Collaborative Filtering*

Polezhaeva E. A.
Moscow, Lomonosov Moscow State University, Faculty of Computational Mathematics and Cybernetics

In modern collaborative filtering applications initial data are typically very large (holding millions of users and items) and come in real time. In this case only incremental algorithms are practically useful. In this paper a new algorithm is proposed based on the symbiosis of Incremental Singular Value Decomposition (ISVD) and Incremental Pearson Correlation (IPC). The algorithm does not require to store the initial data matrix and effectively recalculates user/item profiles and similarity estimates when a new user or a new item appears or a data cell is modified.

Введение

Методы коллаборативной фильтрации (collaborative filtering, CF) используются в рекомендующих системах и системах управления взаимоотношениями с клиентами (CRM) для автоматического формирования персональных предложений. Исходными данными является матрица $Y = (y_{ur})_{n \times d}$, строки которой соответствуют n клиентам, столбцы — d объектам. В зависимости от приложения объектами могут быть товары, услуги, документы, и т. д. Каждая заполненная ячейка матрицы содержит информацию об использовании данным клиентом данного объекта. Это может быть отметка о посещении, выставленный клиентом рейтинг, сумма платежа, и т. д. Задача состоит в том, чтобы для произвольного клиента спрогнозировать оценки предпочтительности объектов по всем незаполненным ячейкам в строке матрицы Y. Предполагается, что для этого выделяется множество клиентов со схожими предпочтениями (коллаборация).

Простые методы CF, основанные на поиске корреляций между клиентами (user-based) или объектами (item-based), неэффективны по времени и по памяти, поскольку они требуют хранения всей матрицы Y. Этих недостатков лишены методы сингулярного разложения SVD [1], неотрицательного матричного разложения NNMF [2] и вероятностного латентного семантического анализа PLSA [3], позволяющие формировать сжатые описания (профили) клиентов и объектов. В современных динамических приложениях к методам CF предъявляется также требование *инкрементности*: метод должен эффективно пересчитывать хранимую информацию в случаях появления (1) нового клиента или объекта и (2) нового значения в ячейке матрицы Y. Обычно в работах по CF рассматривается только один из этих двух типов инкрементности. В данной работе предлагается объединить оба типа инкрементности, используя инкрементное сингулярное разложение (ISVD) и формулу инкрементного вычисления корреляции Пирсона (IPC) [4]. С помощью ISVD оцениваются профили клиентов, затем по этим профилям с помощью IPC вычисляются функции сходства клиентов. Выделяются клиенты, которые наиболее близки клиенту, для которого осуществляется поиск рекомендаций. Находятся наиболее популярные объекты для близких клиентов. Это позволяет отсортировать объекты по значениям рейтинга и выдать топ-список лучших объектов для рекомендации клиенту. В некоторых случаях, описанных ниже, вычисления могут производиться параллельно.

Инкрементное вычисление корреляции Пирсона (IPC)

В коллаборативной фильтрации корреляция Пирсона является одним из наиболее распространённых способов оценивания сходства $\mathrm{sim}(u, v)$ между клиентами u, v. Обычно матрица $Y = (y_{ur})$ является сильно разреженной, и для произвольного клиента u большинство элементов в строке пустые, $y_{ur} = \varnothing$, что обязательно должно учитываться при

Работа выполнена при финансовой поддержке РФФИ, проект № 10-07-00609.

вычислении корреляции Пирсона:

$$\mathrm{sim}(u,v) = \frac{\sum\limits_{r\in R_{uv}} (y_{ur}-\bar{y}_u)(y_{vr}-\bar{y}_v)}{\sqrt{\sum\limits_{r\in R_{uv}} (y_{ur}-\bar{y}_u)^2 \sum\limits_{r\in R_{uv}} (y_{vr}-\bar{y}_v)^2}}, \quad (1)$$

где $R_{uv} = \{r\colon y_{ur} \neq \varnothing,\ y_{vr} \neq \varnothing\}$ — подмножество объектов, которые оценили клиенты u и v, причём, как правило, $|R_{uv}| \ll d$; y_{ur} — рейтинг, данный клиентом u объекту r; $\bar{y}_u, \bar{y}_v$ — средние рейтинги клиентов u и v.

Когда клиент u добавляет новый рейтинг или изменяет имеющийся, должны изменяться и значения сходства между этим клиентом и остальными. Инкрементное вычисление корреляции Пирсона [4] основано на представлении (1) в виде функции от трёх переменных B, C, D, которые вычисляются аддитивно по элементам матрицы Y:

$$A = \mathrm{sim}(u,v) = B/\sqrt{CD};$$
$$B = \sum_{r\in R_{uv}} (y_{ur}-\bar{y}_u)(y_{vr}-\bar{y}_v);$$
$$C = \sum_{r\in R_{uv}} (y_{ur}-\bar{y}_u)^2; \quad D = \sum_{r\in R_{uv}} (y_{vr}-\bar{y}_v)^2.$$

Значения переменных A, B, C, D можно быстро пересчитать как после заполнения ранее пустой ячейки матрицы Y, так и после изменения значения ячейки, по общей формуле

$$A_{\mathrm{new}} = B_{\mathrm{new}}/\sqrt{C_{\mathrm{new}} D_{\mathrm{new}}};$$
$$B_{\mathrm{new}} = B + b; \quad C_{\mathrm{new}} = C + c; \quad D_{\mathrm{new}} = D + d;$$

где величины b, c, d, называемые *инкрементами*, вычисляются по-разному в случаях добавления и изменения ячейки. Рассмотрим эти случаи по отдельности.

Добавление нового рейтинга. Требуется найти сходство между клиентами u и v, когда клиент v добавляет новый рейтинг для объекта r. Возможны два случая:

— клиент u уже оценил r; тогда B, C, D изменяются в соответствии с $\bar{y}'_v$ — новым средним рейтингом v, добавленным рейтингом v для объекта r и новым количеством взаимно оцененных объектов клиентами u и v:

$$b = (y_{vr}-\bar{y}'_v)(y_{ur}-\bar{y}_u) - \sum_{r\in R_{uv}} \delta\bar{y}_v (y_{ur}-\bar{y}_u);$$
$$c = (y_{vr}-\bar{y}'_v)^2 + \sum_{r\in R_{uv}} \delta\bar{y}_v^2 - 2\sum_{r\in R_{uv}} \delta\bar{y}_v (y_{vr}-\bar{y}_v);$$
$$d = (y_{ur}-\bar{y}_u)^2,$$

где
$\bar{y}_u$, $\bar{y}_v$ — средний рейтинг клиентов u и v;
$\bar{y}'_v = \frac{y_{vr}}{m_v+1} + \frac{m_v}{m_v+1}\bar{y}_v$ — новое значение среднего рейтинга активного клиента v;
m_v — число объектов, оцененных клиентом v;
$\delta\bar{y}_v = \bar{y}'_v - \bar{y}_v$ — разность текущего и предыдущего значений среднего рейтинга клиента v;

— клиент u не оценивал r; тогда B, C изменяются в соответствии с новым средним рейтингом v (в этом случае формулы для инкрементов выписываются аналогично).

Изменение рейтинга. Требуется найти сходство между клиентами u и v, когда клиент v модифицирует рейтинг для объекта r. Возможны два случая:

— клиент u уже оценил r, тогда B, C изменяются в соответствии с новым средним рейтингом v и добавленным рейтингом v для объекта r;
— клиент u не оценивал r , тогда B, C изменяются в соответствии с новым средним рейтингом v.

Следующие величины требуется хранить и изменять при работе метода:

— B, C, D для всех пар клиентов u, v;
— m_u — количество объектов, оцененных каждым клиентом u;
— $\bar{y}_u$ — средний рейтинг для каждого клиента u;
— $\bar{y}'_v$ — новое значение среднего рейтинга активного клиента v:
$\bar{y}'_v = \frac{y_{vr}}{m_v+1} + \frac{m_v}{m_v+1}\bar{y}_v$ при добавлении рейтинга,
$\bar{y}'_v = \frac{\delta\bar{y}_v}{m_v} + \bar{y}_v$ при модификации рейтинга;
— $\delta\bar{y}_u = \bar{y}'_u - \bar{y}_u$ — разность текущего и предыдущего среднего значения рейтинга клиента u;
— $\sum\limits_{r\in R_{uv}} y_{ur}$, $\sum\limits_{r\in R_{uv}} y_{vr}$ — сумма рейтингов по объектам, оцененным каждой парой клиентов u, v.

Такое разбиение формулы на части при хранении промежуточных величин повышает эффективность вычисления оценок сходства клиентов и объектов. Важными являются следующие параметры: n' — число клиентов, с которыми данный клиент имеет хотя бы один общий объект, как правило, $n' \ll n$ ($n' > 0$ — условие, позволяющее оценивать сходство); d' — число объектов, не оцененных данным клиентом, но оцененных по крайней мере одним из близких к нему клиентов, как правило, $d' \ll d$ ($d' > 0$ — условие того, что хотя бы один объект может быть рекомендован данному клиенту).

Метод не требует хранения всей матрицы Y. Модификация ячейки матрицы Y требует $O(n'd')$ операций для поддержания матрицы сходств клиентов (не более, чем n' сходств надо модифицировать, и не больше, чем d' объектов просмотреть для формирования рекомендаций клиенту). Для сравнения, в классическом методе CF сложность вычислений будет порядка n^2, так как надо дважды в цикле «обойти» всех клиентов (рассмотреть все пары клиентов), и потом еще объекты, оцененные обоими клиентами.

Инкрементное сингулярное разложение (ISVD)

Второй подход связан с сингулярным разложением, которое позволяет представить матрицу Y в виде произведения двух ортогональных матриц U и R и диагональной матрицы S, так что верны равенства $USR^\intercal = Y,\ U^\intercal YR = S$. Диагональные элементы S называются сингулярными числами, а столбцы U и R — левыми и правыми сингулярными векторами, соответственно. Оставим только r наибольших сингулярных чисел в S, у U и R оставим только столбцы, отвечающие этим сингулярным числам. Результат после прореживания матриц $U'S'R'^\intercal \approx Y$ есть наилучшая аппроксимация ранга r матрицы Y в смысле наименьших квадратов. Нас будет интересовать именно такое разложение.

Постановка задачи

Рассмотрим такие случаи модификации данных, добавления и удаления строк и столбцов к матрице Y, при которых происходит изменение её ранга на 1. Пусть есть два вектор-столбца a и b и известное SVD $USR^T = Y$. Требуется найти SVD матрицы $Y + ab^T$. Бинарный вектор b определяет, какие столбцы должны быть модифицированы, а вектор a определяет, что требуется получить: добавить или удалить значения, либо модифицировать значения в выбранных строках и столбцах.

Например, добавление столбца можно описать так: $US[R^\intercal\ \ 0] = [Y\ \ 0]$ — известное разложение, $U'S'R'^\intercal = [Y\ \ c]$ — требуемое разложение, $a = c$, $b = [0, \ldots, 0, 1]^\intercal$.

Пусть $USR^\intercal = Y$ есть SVD ранга r матрицы $Y_{n\times d}$, у которой большинство элементов пустые, где $U^\intercal U = R^\intercal R = I$. Покажем, как модифицировать U, S, R в более общем случае, чтобы получить SVD $Y + AB^\intercal$, где A и B имеют c столбцов ($A_{n\times c}$,$B_{d\times c}$).

Получение нового SVD

Нас интересует SVD [1]:

$$Y + AB^\intercal = [U, A] \begin{bmatrix} S & 0 \\ 0 & I \end{bmatrix} [R, B]^\intercal \qquad (2)$$

Пусть P — ортогональный базис пространства столбцов $\tilde{A} = (I - UU^\intercal)A = A - UU^\intercal A$. Применим QR-разложение к $\tilde{A}$: $\tilde{A} = PR_A$, где P — ортонормированная, а R_A — верхняя треугольная матрица, $R_A = P^\intercal(I - UU^\intercal)A$.

Итак, запишем QR-разложение матрицы $[U, A]$:

$$[U, P] \begin{bmatrix} I & U^\intercal A \\ 0 & R_A \end{bmatrix} = [U, A] \qquad (3)$$

Разложение может быть эффективно вычислено, в частности, с помощью модифицированной ортогонализации Грама–Шмидта [5].

Аналогично, пусть Q — ортогональный базис пространства столбцов $B - RR^\intercal B$, тогда $QR_B = (I - RR^\intercal)B$, где Q — ортонормированная, а R_B — верхняя треугольная матрица, $R_B = Q^\intercal(I - RR^\intercal)B$.

Представим матрицу $Y + AB^\intercal$ в виде произведения трёх матриц:

$$\begin{aligned} Y + AB^\intercal &= \\ &= [U, A] \begin{bmatrix} I & U^\intercal A \\ 0 & R_A \end{bmatrix} \begin{bmatrix} S & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & R^\intercal B \\ 0 & R_B \end{bmatrix} [R, Q]^\intercal \\ &= [U, P] \underbrace{\left(\begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U^\intercal A \\ R_A \end{bmatrix} \begin{bmatrix} R^\intercal B \\ R_B \end{bmatrix}^\intercal \right)}_{K} [R, Q]^\intercal; \end{aligned}$$

$$K = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} (U^\intercal A)(B^\intercal R) & (U^\intercal A)R_B^\intercal \\ R_A(B^\intercal R) & R_A R_B^\intercal \end{bmatrix}.$$

Пусть $U'S'R'^\intercal$ — SVD ранга $(r + c)$ матрицы K. Диагонализируя K, $U'^\intercal K R' = S'$, получаем новые матрицы U', S', R'. Тогда модификация SVD ранга r до ранга $(r + c)$ выглядит так:

$$Y + AB^\intercal = \big([U\ P]U'\big)S'\big([R\ Q]R'\big)^\intercal. \qquad (4)$$

Однако, вместо того, чтобы оперировать с матрицами больших размеров, вводится SVD разложение матрицы $Y + AB^\intercal$, состоящее из пяти матриц:

$$U_{n\times r}U'_{r\times r}S_{r\times r}R'^\intercal_{r\times r}R^\intercal_{r\times d} \qquad (5)$$

где UU', RR', U, U' — ортонормированные матрицы, U' и R' — матрицы, позволяющие сохранять диагональность S, поэтому вычисления производятся быстро.

Если добавляется одна строка или столбец, то модификации можно производить следующим образом: для измененного SVD (a — вектор размерности n, b — вектор размерности d) используем модифицированную ортогонализацию Грама–Шмидта:

$$\begin{aligned} z &= U^\intercal a; & p &= a - Uz; \\ w &= R^\intercal b; & q &= b - Rw; \end{aligned}$$

Тогда верно:

$$K = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} zw^\intercal & z\|q\| \\ \|p\|w^\intercal & \|p\|\|q\| \end{bmatrix}.$$

Матрица K имеет очень простой вид, что обуславливает возможность ее эффективной по времени диагонализации и получения нового SVD. После получения нового SVD к матрице US применяется формула инкрементного вычисления корреляции Пирсона, и находится сходство между клиентами. При добавлении объекта такой процесс последователен и требует $O((nd + d)r)$ операций, но добавление клиента позволяет использовать параллельные вычисления.

Алгоритм 1. Совместный ISVD и IPC.

Вход: $Y \in \mathbb{R}^{n\times d}$; $a \in \mathbb{R}^n$; $b \in \mathbb{R}^d$;
Выход: p_{ur};

1: **если** модификация одной ячейки матрицы **то**
2: используем IPC для Y, сохраняем в массив $\delta y_{ij} = y'_{ij} - y_{ij}$ (разность текущего и предыдущего значения при изменении одной ячейки матрицы Y) , при этом возможно параллельно с IPC найти SVD с помощью ISVD;
3: **иначе если** новый клиент **то**
4: $u_{\text{new}} = y_{\text{new}} RS^{-1}$. Находим $\text{sim}(u, v)$ для US, при этом возможно параллельно найти SVD с помощью ISVD;
5: **иначе если** новый объект **то**
6: Применяем ISVD к матрице Y, затем находим $\text{sim}(u, v)$ для $UU'S$;
7: Применяем (6).

ISVD при добавлении клиента

При добавлении нового клиента возможно улучшение по времени: эффективное вычисление новой строки матрицы U, отвечающей добавленному клиенту, позволяет применять формулу вычисления корреляции Пирсона для US параллельно с получением ISVD. Используя результаты [6], можно при добавлении нового клиента u быстро найти новую строку для матрицы U, характеризующую данного нового клиента, зная только вектор рейтингов объектов клиентом u (новую строку Y), а также матрицы предыдущего разложения SVD:

$$u_{\text{new}} = y_{\text{new}} RS^{-1},$$

где u_{new} — новая вектор-строка матрицы U, y_{new} — вектор рейтингов нового клиента (новая строка в матрице Y). Таким образом, имея новое U, можем найти сходство клиентов через профили, применяя IPC к US, а параллельно находить новое SVD. В результате добавление клиента требует $O((nd + n)r)$ операций.

Общий алгоритм ISVD и IPC

Для того, чтобы дать новый топ-список рекомендуемых объектов для клиента, будем использовать функцию сходства клиентов, основанное на ISVD и IPC. Чтобы выделить топ-список объектов, будем использовать значение средней абсолютной ошибки (MAE — Mean Absolute Error):

$$p_{ur} = \bar{y}_u + \frac{\sum\limits_{v\in U} \text{sim}(u, v)(y_{vr} - \bar{y}'_u)}{\sum\limits_{v\in U} \text{sim}(u, v)}. \qquad (6)$$

Далее сортируем объекты по значениям рейтинга и получаем топ-список объектов.

Приведенные выше рассуждения приводят к алгоритму совместного инкрементного сингулярного разложения (ISVD) и инкрементного вычисления корреляции Пирсона (IPC). Псевдокод приведён в Алгоритме 1.

Выводы

Предложен метод, позволяющий эффективно хранить информацию, а также эффективно находить рекомендации для клиентов через оценку их сходства, используя икрементное вычисление корреляции Пирсона и инкрементное сингулярное разложение. При этом объем хранимых данных в случае исходной матрицы Y размера $n \times d$ ранга r составляет $O(n + kn' + (n + d)r)$, где n' — число клиентов, с которыми данный клиент имеет хотя бы один общий объект (обычно $n' \ll n$); k — число клиентов, имеющих хотя бы с одним другим клиентом общий объект. Добавление клиента требует $O((nd + n)r)$ операций, добавление объекта — $O((nd + d)r)$ операций, модификация ячейки матрицы Y — $O(n'd')$ операций.

В будущем предполагается разработать и исследовать модификацию инкрементного метода, позволяющую работать с сильно разреженными матрицами.

Литература

[1] *Brand M.* Fast Low-rank modifications of the thin singular value decomposition // Linear Algebra and Its Applications. — 2006. — Т. 415, № 1. — С. 20–30.

[2] *Lee D.D., Seung H.S.* Learning the parts of objects by nonnegative matrix factorization // Nature. — 1999. — Т. 401(6755), — С. 788–791.

[3] *Leksin V.A.* Symmetrization and overfitting in probabilistic latent semantic analysis // Pattern Recognition and Image Analysis. — 2009. — Т. 19, № 4. — С. 565-574.

[4] *Papagelis M.* Incremental collaborative filtering for highly-scalable recommendation algorithms // ISMIS, Springer. — 2005. — Т. 3488, — С. 553-561.

[5] *Golub G.* Matrix Computations. — Johns Hopkins University Press, 1996. — 728 с.

[6] *Furnas G.* Information retrieval using a singular value decomposition model of latent semantic structure // In Proc. ACM SIGIR Conf, 1988. — С. 465–480.

Динамическое обучение распознаванию статистических таблиц*

Кудинов П. Ю., Полежаев В. А.
pkudinov@gmail.com
Москва, ВЦ имени А. А. Дородницына РАН

При создании единого хранилища социально-демографических и экономических данных возникает задача его постоянного пополнения разнородными по структуре и форматам таблицами, получаемыми из различных источников. Для полуавтоматического решения данной задачи предлагается использовать алгоритмы динамического обучения классификации. Описываются требования к алгоритмам классификации и приводится анализ некоторых алгоритмов, которые могут использоваться для обработки таблиц.

On-line learning in statistical table processing*

Kudinov P., Polezhaev V.
Dorodnicyn Computing Centre of RAS, Moscow, Russian Federation

When creating unified repository of socio-demographic and economic data there arises a problem of its continuous replenishment by tables received from different sources and having heterogeneous structure and format. A semi-automated solution based on on-line learning is proposed in this paper. Requirements to on-line learning algorithms are described and some algorithms are reviewed and tested on real sets of tables.

Анализ социально-демографических и экономических статистических данных является неотъемлемой частью многих исследований в экономике, социологии, других гуманитарных областях. Большой проблемой на сегодняшний день является разрозненность данных, представленных как в Интернете, так и в официальных источниках. Не существует единого удобного, надёжного, свободно доступного источника таких данных. Имеющиеся источники представляют данные в виде таблиц различных и плохо совместимых структур и форматов. Отсутствуют строгие правила, по которым составляются эти таблицы. Зачастую данные неполны, неточны и противоречат друг другу.

Таким образом, актуальна задача создания системы, с помощью которой можно было бы собирать статистические данные из разных источников, накапливать их и анализировать. Одной из основных функций такой системы должна быть выдача данных по запросам пользователей, в естественной табличной форме, с указанием первоисточников. В отличие от традиционных поисковых систем, организация поиска по таблицам требует нетривиальной предварительной обработки данных. После того, как новая таблица найдена и представлена в виде совокупности ячеек, необходимо выполнить её семантический анализ — для каждой ячейки определить единственно правильный способ записи её значения в хранилище, то есть, фактически, определить смысл ячейки.

Основные этапы выделения статистических данных из таблиц произвольной структуры были описаны в [1]. В данной работе отдельные подзадачи семантического анализа таблиц ставятся как задачи динамического обучения классификации [2]. Предполагается, что семантический анализ выполняется в полуавтоматическом режиме: таблица сначала размечается автоматически, несколькими алгоритмами классификации. Затем эта разметка предъявляется оператору (эксперту, выступающему в роли «учителя»), который при необходимости исправляет ошибки. Факты исправлений пополняют обучающую выборку и происходит дообучение алгоритмов. По мере увеличения обучающей выборки оператор тратит всё меньше и меньше времени на исправления. Это радикальное снижение трудозатрат по сравнению с обычным, нединамическим, подходом к обучению, когда оператору приходится размечать все поступающие таблицы.

Важным требованием к процедуре динамического обучения является *корректность* — алгоритмы должны дообучаться так, чтобы результат их применения к ранее проверенным таблицам не изменялся. Отметим, что требование корректности не предъявляется к большинству известных методов динамического обучения, что накладывает существенные ограничения на выбор метода, вынуждает строить корректные модификации известных методов и исследовать их корректность.

Основные понятия и определения

Таблица состоит из совокупности *ячеек*. Каждая ячейка может иметь один из трёх типов: ячейка данных, ключ, или пустая (неинформативная). *Ячейка данных* содержит одно значение, как правило, числовое. Для каждой ячейки данных по таблице должен быть определён, во-первых, набор ключей, во-вторых, *единицы измерения*.

Ключ представляет собой текстовую строку, которая относится к множеству ячеек данных, и задаёт (возможно, лишь частично) смысл этих ячеек.

Работа выполнена при финансовой поддержке РГНФ, проект № 08-02-12104в, и РФФИ, проект № 10-07-00609.

Распределение численности занятых в экономике регионов Российской Федерации по возрастным группам в 2000 г. (в процентах)

	Всего	в том числе в возрасте, лет						Средний возраст, лет
		до 20	20-29	30-39	40-49	50-59	60-72	
Российская Федерация	**100**	**2,0**	**21,5**	**27,2**	**30,3**	**14,1**	**5,0**	**39,3**
Центральный федеральный округ	**100**	**1,6**	**20,0**	**26,6**	**30,1**	**15,7**	**6,1**	**40,1**
Белгородская область	100	1,8	19,8	28,4	30,5	12,6	6,9	39,9
Брянская область	100	1,9	22,8	27,7	30,7	12,3	4,6	38,8
Владимирская область	100	2,2	21,0	26,5	30,6	14,4	5,2	39,4
Воронежская								

Рис. 1. Пример исходной таблицы простой структуры.

Набор ключей задаёт смысл ячейки данных полностью, то есть позволяет однозначно определить способ записи значения ячейки в хранилище.

На Рис. 1 представлен пример исходной таблицы простой структуры с одним блоком ячеек данных (светло-серый) и двумя блоками ключей (тёмно-серые). Значение 19,8, находящееся на пересечении 5-й строки и 4-го столбца, имеет следующий набор ключей: показатель «распределение численности занятых в экономике регионов Российской Федерации»; возрастная группа «20–29 лет»; период «2000 год»; территория «Белгородская область». Единицей измерения является «процент».

Как видно из примера, каждый ключ характеризуется типом, таким как «показатель», «возрастная группа», «период», «территория», «отрасль», и т. д. Число типов относительно невелико, тогда как число различных ключей может быть на порядки больше. Для каждого типа ключей строится словарь возможных значений ключей. В процессе обучения системы могут создаваться новые ключи, типы ключей и единицы измерения.

Задачи обучения

Семантический анализ каждой ячейки таблицы сводится к решению следующих подзадач:

1) распознавание типа ячейки;
2) распознавание структуры таблицы (взаимного расположения блоков данных и ключей);
3) поиск местоположения ключей, относящихся к ячейке данных;
4) выделение единиц измерения ячейки данных;
5) выделение периода времени;
6) распознавание типа ключа, либо принятие решения о создании нового типа ключа;
7) поиск подходящего ключа в словаре, либо принятие решения о синонимии ключей, либо принятие решения о создании нового ключа.

Заметим, что каждая из этих задач в принципе могла бы решаться чисто программистскими эвристическими приёмами. Однако необходимость постоянной адаптации такой программы к новым таблицам рано или поздно привела бы к неуправляемому росту её сложности и невозможности дальнейшего развития. Рассмотрим некоторые постановки задач обучения классификации.

Распознавание типа ячейки. Число классов равно трём: данные, ключ, пусто. Объектами являются ячейки таблицы. Признаками являются: доля цифровых последовательностей символов; доля буквенных последовательностей; относительный номер строки и столбца; длина строки; длина текста первой ячейки в строке. Результат классификации показывается оператору в виде цветной раскраски таблицы, см. Рис. 1.

Распознавание структуры таблицы. Таблицы различаются по структуре и типу связей между ячейками. Учёт этих различий необходим для создания оптимизированных процедур обработки таблиц каждого из типов. Предлагается выделять следующие типы таблиц:

1) простая таблица или таблица со сложными шапками;
2) таблица с супер-строками или супер-столбцами;
3) таблица с нетривиальным расположением ключей, относящихся к ячейкам с данными;
4) таблица, построенная в результате конкатенации нескольких таблиц.

При анализе таблицы генерируются следующие признаки: число colspan в шапках; число rowspan в шапках; число супер-строк наличие сдвигов внутри ячеек; наличие ячеек-указателей на другие ячейки («в том числе», «из них», двоеточия в конце текста); количество связанных множеств ячеек с данными; количество связанных множеств ячеек с ключами.

Выделение периода времени и единиц измерения. Для каждого числа в таблице должны быть определены период времени, к которому данное значение относится, и единица измерения данного значения. Обычно период времени и единица измерения выражены в тексте одного из ключей и отделены каким-либо способом от основного текста, например точкой с запятой, скобками и т. п. Эта информация должна быть удалена из текста прежде, чем будет осуществлен поиск наиболее близкого ключа в словаре. Оператор системы имеет возможность выделять единицы измерения даты в текстах ключей и, тем самым, обучать систему их расположению. По полученным данным строится обучающая выборка, состоящая из двух классов — даты и единицы измерения, а в качестве признаков используются следующие: небуквенные символы-разделители слева и справа от выделения; количество слов; количество существительных; количество прилагательных; количество непрерывных последовательностей цифр; часть речи первого слова; число сокращений.

Алгоритмы динамического обучения

При динамическом обучении (on-line learning) объекты появляются по одному, причём каждый объект обрабатывается только один раз. Динамическое обучение состоит из последовательности чередующихся шагов классификации и дообучения. На стадии классификации очередного объекта алгоритму не известно значение целевого признака, однако непосредственно после классификации ему передаётся правильный ответ, который используется для модификации параметров алгоритма (дообучения). Алгоритмы классификации, используемые для решения наших задач, должны удовлетворять требованию корректности, то есть при последующих модификациях они не должны делать ошибок на ранее классифицированных объектах.

Для сравнения были выбраны три алгоритма: ближайших соседей, RISE [3] и ITI (Incremental Tree Inducer) [4, 5]. Выбор этих алгоритмов обусловлен тем, что они основаны на запоминании обучающей выборки, что позволяет обеспечить выполнение требования корректности.

Были проведены эксперименты на реальных данных для задачи распознавания типа ячейки. Обучающая выборка состояла из 20 разнообразных таблиц, в каждой из которых было около 100 ячеек. Каждый эксперимент состоял из серии запусков, перед каждым из которых происходило случайное перемешивание таблиц, а затем ячеек в них. В каждом запуске небольшая часть исходной выборки использовалась для начального обучения алгоритма. Остальные объекты подавались алгоритмам последовательно. После классификации очередного объекта алгоритму давался правильный ответ и происходило его дообучение. Затем проверялась классификация всех ранее классифицированных объектов, чтобы удостовериться, что требование корректности выполнено.

Алгоритм k ближайших соседей основан на полном запоминании всей выборки. Обучение алгоритма заключается в добавлении нового объекта в список известных. При классификации происходит поиск k ближайших объектов и выбирается доминирующий класс. В качестве меры близости использовалось евклидово расстояние без взвешивания признаков. Очевидным недостатком данного алгоритма является скорость его работы. Результат эксперимента для этого алгоритма представлен на Рис. 2. Данный алгоритм позволяет получить неплохое значение ошибки на новых объектах и не делает ошибок на старых объектах.

Алгоритм RISE строит правила из объектов обучающей выборки. Классификация заключается в поиске ближайшего правила и использовании его класса в качестве ответа с помощью функции расстояния между правилом R, заданным в виде

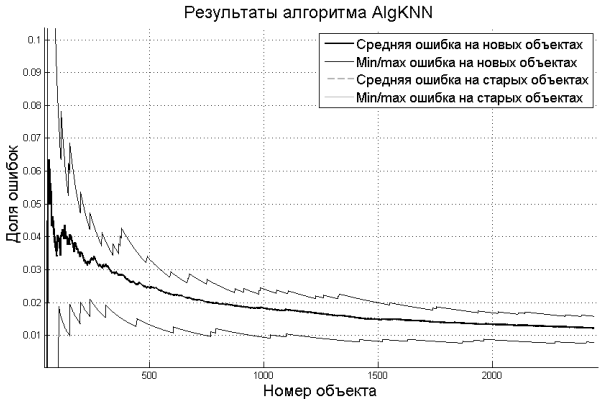

Рис. 2. Результат работы алгоритма KNN.

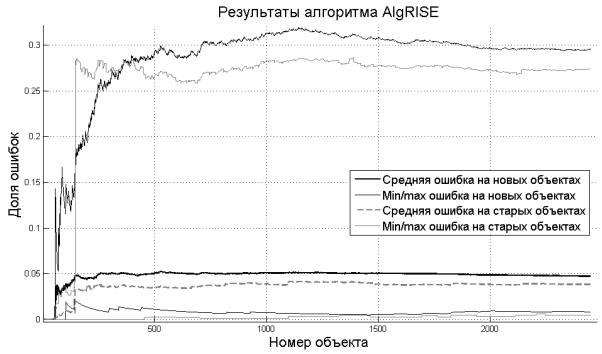

Рис. 3. Результат работы алгоритма RISE.

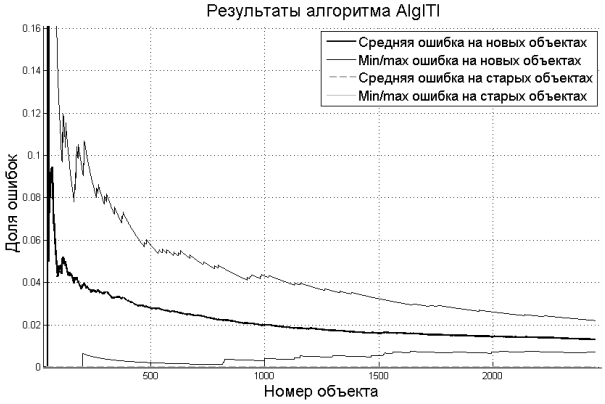

Рис. 4. Результат работы алгоритма ITI.

конъюнкции предикатов $r_1 \wedge \cdots \wedge r_N$, и объектом $E = (e_1, \ldots, e_N)$:

$$\Delta(R, E) = \sum_{i=1}^{N} \delta^2(i),$$

где $\delta(i)$ — функция расстояния по i-му признаку:

$$\delta(i) = \begin{cases} 0, & r_i = true; \\ \text{orddist}(i), & i\text{-й признак номинальный}; \\ \text{numdist}(i), & i\text{-й признак числовой}. \end{cases}$$

Предикатом для номинального признака является индикатор принадлежности некоторому подмноже-

ству допустимых значений S, а для числового — индикатор принадлежности отрезку $[a, b]$. Функция orddist(i) принимает значения 0 или 1 в зависимости от принадлежности множеству S, а функция numdist(i) равна расстоянию до отрезка $[a, b]$. Сначала из каждого объекта $e = (e_1, \ldots, e_N)$ обучающей выборки порождаются правило, состоящее из предикатов вида $r_i(x) = [x_i = e_i]$, если i-й признак номинальный и $r_i(x) = [x_i \in [e_i - \varepsilon, e_i + \varepsilon]]$, $\varepsilon > 0$, если i-й признак числовой. Далее правила обобщаются до ближайшего с точки зрения расстояния $\Delta(R, E)$ непокрытого объекта. Для каждого номинального признака в предикат добавляется значение этого признака на объекте E, а для числового — отрезок предиката расширяется таким образом, чтобы значение признака объекта E попадало в интервал. После этого происходит удаление тех правил, которые полностью поглащаются хотя бы одним другим правилом.

На Рис. 3 представлен результат эксперимента для алгоритма RISE. Видно, что он делает ошибки на старых объектах, поэтому его использование в системе затруднительно.

Алгоритм ITI строит бинарное решающее дерево. В узлах дерева хранятся простые предикаты для одного признака. Для номинальных признаков — это проверка на равенство одному из его значений, а для числовых — неравенство вида $[x_i < a]$. В листьях дерева накапливаются объекты одного класса. Построение дерева инкрементно: при поступлении нового объекта происходит его классификация с использованием дерева; когда объект достигает листа дерева, осуществляется проверка на совпадение класса, обозначенного в листе, с классом объекта. При совпадении новый объект добавляется в лист, иначе лист преобразуется в узел. Предикат p нового узла выбирается на основе множества объектов X, хранящихся в листе, так, чтобы он был наиболее информативен с точки зрения энтропийного критерия информативности

$$I_p(X) = \frac{|X_+|}{|X|} I(X_+) + \frac{|X_-|}{|X|} I(X_-),$$

где X_+ и X_- — множества, объекты которых предикат p выделяет и не выделяет соответственно. Для K классов $C_1, \ldots, C_K$ информативность множества S задается по формуле

$$I(S) = -\sum_{j=1}^{K} \frac{|C_j \cap S|}{|S|} \log \frac{|C_j \cap S|}{|S|}.$$

Когда условие установлено, образуются два листа, исходящих из нового узла, в которые записываются множества объектов X_+ и X_-.

Результаты эксперимента с алгоритмом ITI представлены на Рис. 4. На графике видно, что значение ошибки на новых объектах монотонно падает, а график ошибки на старых объектах совпадает с осью абсцисс.

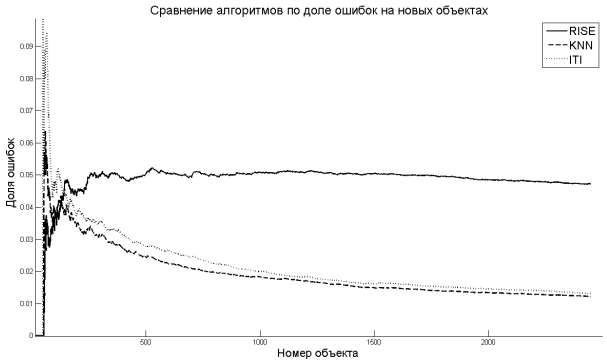

Рис. 5. Сравнение алгоритмов по числу ошибок на новых объектах.

Сравнение алгоритмов. На Рис. 5 приведены графики ошибок на новых объектах для всех алгоритмов. Худший результат показал алгоритм RISE — среднее значение ошибки стабилизировалось на уровне 5%.

Значение ошибки на новых объектах у алгоритмов kNN и ITI было примерно одинаковым, на уровне 2%. При этом алгоритм ITI показал хорошую скорость работы, и это позволяет использовать его в дальнейших исследованиях и разработке системы распознавания таблиц.

Заключение

В статье рассмотрены задачи динамического обучения, решаемые в системе распознавания статистических таблиц. Описана процедура формирования обучающей выборки и классификации, приведены примеры задач динамического обучения. Сравнительный анализ трёх алгоритмов динамического обучения показал, что целесообразность использования алгоритма ITI.

Литература

[1] *Кудинов П. Ю.* Задача распознавания статистических таблиц // Математические методы распознавания образов-14. М.: МАКС Пресс, 2009. — С. 552–555.

[2] *Blum A.* On-Line Algorithms in Machine Learning // Carnegie Mellon University, Pittsburgh PA. — 15213.

[3] *Domingos P.* Rule Induction and Instance-Based Learning: A Unified Approach // Department of Information and Computer Science University of California, Irvine, California 92717, U.S.A.

[4] *Utgoff P. E.* An Improved Algorithm for Incremental Induction of Decision Trees // 1994, Department of Computer Science, University of Massachusetts, Amherst, MA 01003.

[5] *Utgoff P. E., Berkman N. C., Clouse J. A.* Decision tree induction based on efficient tree restructuring // Machine Learning. — 1997. — No. 29, Pp. 5–44.

Системы тестирования алгоритмов машинного обучения: MLcomp, TunedIt и Полигон*

Лисица А. В., Воронцов К. В., Ивахненко А. А., Инякин А. С., Синцова В. В.
lisitsa@forecsys.ru, voron@ccas.ru, {ivahnenko, inyakin, sintsova}@forecsys.ru
Москва, Вычислительный Центр РАН, ЗАО «Форексис»

«Полигон» — это бесплатная веб-система для тестирования алгоритмов классификации на реальных и модельных данных, разработанная в ВЦ РАН и ЗАО «Форексис». Цель системы — предоставить исследователям и прикладным специалистам удобную площадку для анализа алгоритмов и задач классификации, а также стандартизировать методику тестирования. Аналогичные цели преследуют системы MLcomp и TunedIt. В докладе приводится сравнительный анализ этих трех систем.

Online services for testing machine learning algorithms: MLComp, TunedIt, and Poligon*

Lisitsa A. V., Vorontsov K. V., Ivahnenko A. A., Inyakin A. S., Sintsova V. V.
Dorodnicyn Computing Centre of RAS, Moscow, Russia; Forecsys, Moscow, Russia

Poligon is a free web-based system developed in Dorodnicyn Computing Centre of RAS and Forecsys company for testing machine learning algorithms on real or model data. Poligon provides a convenient platform for the analysis of classification algorithms and tasks as well as standardize the testing procedure. MLcomp and TunedIt have analogous purposes. This paper gives the comparative study of three systems.

Создание новых алгоритмов классификации является сложной задачей. Для того, чтобы алгоритм был востребован, его эффективность должна быть подтверждена экспериментами на реальных данных. Эксперименты должны быть легко воспроизводимыми, чтобы любой ранее полученный результат можно было перепроверить, и стандартизованными, чтобы результаты, полученные разными исследователями, были сопоставимы.

В машинном обучении точной воспроизводимости достичь довольно сложно, так как это требует соблюдения ряда условий.

1. *Идентичность реализации алгоритма.* Авторы алгоритма не всегда предоставляют его реализацию, ограничиваясь описанием идеи. Если же реализация находится в свободном доступе, то трудно гарантировать, что во всех работах используется одна и та же версия. Получение программы или исходных кодов затруднено или невозможно для коммерческих алгоритмов.
2. *Идентичность методики тестирования.* Описания методики кросс-валидации часто поверхностны и не содержат таких важных деталей, как стратификация выборки по классам или по признакам. Более того, точное воспроизведение результатов эксперимента невозможно без знания множества разбиений выборки на обучение и контроль, которое, как правило, генерируется случайным образом и не запоминается.
3. *Идентичность исходных данных.* Хотя данные, как правило, берутся из одного источника (чаще всего из репозитория UCI [4]), детали предварительной обработки (нормировка, заполнение пропусков, и т. д.) тоже часто опускаются.

Если полученные результаты расходятся, то причиной может быть различие и в реализации алгоритмов, и в методике тестирования, и в данных.

Попытки создания систем для автоматизации экспериментов в машинном обучении неоднократно предпринимались в предыдущие десятилетия, но наталкивались на технологические и организационные трудности. Развитие web-технологий привело к созданию в последние годы нескольких общедоступных систем:

— `poligon.MachineLearning.ru` в России (2008);
— `www.TunedIt.org` в Польше (2008);
— `www.MLComp.org` в США (2009).

Поскольку цели этих систем схожи, их функциональные возможности также схожи. Каждая система имеет пополняемый репозиторий задач и алгоритмов. Пользователь системы может ограничивать доступ к добавленной им задаче или алгоритму. Каждая система использует для тестирования алгоритмов методику кросс-валидации, основанную на многократном разбиении исходной выборки данных на «обучение» и «контроль», и позволяет вычислять среднюю ошибку на контроле для любой пары ⟨задача, алгоритм⟩. Каждая система имеет веб-интерфейс для отображения результатов тестирования алгоритмов.

Все системы одинаково определяют круг своих пользователей. Во-первых, это разработчики алгоритмов, которые получают возможность сравни-

Работа выполнена при финансовой поддержке РФФИ (проекты № 10-07-00673, № 08-07-00422) и программы ОМН РАН «Алгебраические и комбинаторные методы математической кибернетики и информационные системы нового поколения».

вать свои алгоритмы со стандартными на представительном наборе реальных задач. Во-вторых, это прикладные специалисты, которые получают возможность выбирать алгоритмы, наиболее подходящие для конкретной задачи, особо не вникая в область машинного обучения В-третьих, все системы возникли как университетские проекты и активно используются в учебном процессе и для организации соревнований по анализу данных. В частности, система Полигон используется на кафедре «Интеллектуальные системы» ФУПМ МФТИ и в Школе анализа данных[1] Яндекс.

Однако имеются и существенные различия в том, где хранятся и как исполняются алгоритмы, какие допускаются типы алгоритмов, как строится множество разбиений, какие характеристики качества вычисляются, и в каком виде выдаются результаты тестирования.

Система TunedIt

Система TunedIt начала разрабатываться в 2008 году. Ее основатель и идейный вдохновитель — Marcin Wojnarski (все началось с его диссертационного проекта; сейчас проект поддерживается Варшавским университетом, а также Технологическим инкубатором Варшавского университета).

Система состоит из трех блоков — «базы знаний», «репозитория задач и алгоритмов» и «тестировщика» (*TunedTester*).

TunedTester позволяет запускать алгоритмы и выгружать результаты тестирования на сервер системы. Этот компонент работает локально, поэтому пользователь должен загрузить и установить на свой компьютер программу с сайта www.tunedit.org. Все необходимые для тестировании задачи и алгоритмы загружаются на компьютер пользователя автоматически при запуске тестирования.

Алгоритмы, используемые *TunedTester*'ом должны быть написаны на языке JAVA. Компонент работает с алгоритмами из библиотек WEKA[2], Rseslib[3] и Debellor[4]. Для добавления алгоритма надо встроить его в одну из этих трех библиотек.

В системе есть стандартная процедура тестирования — разбиение задачи на обучение и контроль (случайное, 70/30). Качество алгоритма оценивается как доля ошибок на контроле. Пользователь может дополнить систему собственной процедурой тестирования (*evaluation procedure*), использующей произвольные разбиения задачи и выдающей на выходе любую другую скалярную величину.

Один запуск *TunedTester*'а заключается в выборе процедуры тестирования, алгоритма и задачи. Результаты тестирования могут быть опубликованы в «базе знаний» на сайте системы. «База знаний» предоставляет интерфейс для сравнения результатов различных алгоритмов. Сравнение возможно только в контексте одной задачи и одной процедуры тестирования. Для каждого алгоритма указывается количество запусков процедуры тестирования, среднее значение и стандартное отклонение данной оценки качества на данной задаче. Для стандартной процедуры тестирования оценкой качества является доля ошибок на контроле.

Подход TunedIt имеет следующие ограничения.

1. Ограничение на язык программирования (JAVA).
2. Отсутствуют механизмы контроля мета-параметров, влияющих на процедуру обучения; например, оценка качества работы SVM может вычисляться при различных ядрах.
3. На выходе процедуры тестирования может быть только скалярная величина, что ограничивает возможности детального анализа.
4. Для тестирования алгоритма необходимо иметь его исполняемый код (*java*-класс), что невозможно в случае коммерческих алгоритмов.

Система MLComp

Система MLComp появилась в конце 2009 года. Её создали аспиранты университета Беркли Jacob Abernethy и Percy Liang. Вычислительные мощности системы расположены в «облаках» Amazon Elastic Compute Cloud[5]. Система предлагает автоматизацию тестирования различных методов машинного обучения: алгоритмов классификации, коллаборативной фильтрации, анализа текста и др.

В MLComp другая схема работы с алгоритмами — алгоритмы исполняются не на стороне пользователя, а на сервере системы. Для тестирования алгоритма пользователь загружает свою программу на сайт и запускает её выполнение (run) на одной из задач. Программа выполняется в среде Linux. Система не накладывает ограничений на язык программирования. Так как алгоритмы выполняются на одном сервере и имеют одинаковые вычислительные ресурсы, в рамках MLComp можно сравнивать скорости работы алгоритмов.

В плане процедур тестирования MLComp значительно уступает TunedIt. Проблема заключается в выбранном подходе к тестированию — формат задачи требует явного указания тестовой и контрольной подвыборок, и эта информация хранится в данных задачи. Все тесты алгоритмов на задаче происходят на одном и том же разбиении, что делает невозможным статистически надежную оценку характеристик качества работы алгоритмов.

[1] http://shad.yandex.ru/

[2] http://www.cs.waikato.ac.nz/ml/weka/

[3] http://rseslib.mimuw.edu.pl/

[4] http://www.debellor.org/

[5] http://aws.amazon.com/ec2/

Подход MLComp имеет следующие ограничения.

1. Только одно разбиение задачи на обучение и контроль, в результате низкая статистическая надежность получаемых оценок.
2. Для тестирования алгоритма необходимо передать разработчикам системы его исполняемый код (программу), что невозможно в случае коммерческих алгоритмов.
3. Ограничение на операционную систему (Linux).

Система Полигон

Система Полигон была анонсирована в 2007 году на конференции ММРО-13 [1] и начала функционировать в 2008 году. Она имеет несколько существенных отличий от предыдущих систем.

Во-первых, это распределённая система. Каждый алгоритм запускается на компьютере пользователя (как правило, автора алгоритма) и автоматически выполняет задания, поступающие от сервера Полигона. Задания формируются исходя из того, какие отчёты запрашивают другие пользователи. Таким образом, каждый пользователь, предоставляющий системе свой алгоритм, добровольно выделяет часть своего вычислительного ресурса, но сохраняет полный контроль над программой и исходным кодом алгоритма. Программа может быть реализована на любом языке, поддерживающем web-сервисы. Благодаря этим архитектурным особенностям к системе можно подключать коммерческие версии алгоритмов.

Во-вторых, в Полигоне существенно глубже проработана методика тестирования [1, 2]. Для каждой задачи хранится и используется только одно представительное множество разбиений, что обеспечивает воспроизводимость и статистическую надёжность результатов. Наряду со стандартной характеристикой — частотой ошибок на контроле — вычисляется большое количество скалярных и графических характеристик для детального анализа качества алгоритмов.

Информация, полученная в ходе тестирования, сохраняется в базе данных системы и в дальнейшем может многократно использоваться для выдачи отчётов пользователям.

Полигон оперирует с четырьмя типами объектов: *задача*, *разбиение*, *алгоритм* и *статистика*.

Репозиторий задач классификации.

Задача классификации Z описывается в Полигоне следующим набором данных:

— матрица $F = (F_{ij}) \in \mathbb{R}^{L\times n}$ значений j-го признака на i-ом объекте;
— вектор $y = (y_i)_{i=1}^{L} \in Y^L$ правильных ответов;
— матрица $C = (C_{uv}) \in \mathbb{R}^{m\times m}$ стоимости потерь при ошибочном соотнесении объекта класса u к классу v ($|Y| = m$);
— вектор информации $I = (I_j)_{j=1}^{n}$ о типах признаков (номинальный, вещественный).

Задачи могут быть загружены в формате ARFF[6], либо во внутреннем формате Полигона — в виде нескольких csv-файлов с данными. Пользователь, добавивший задачу в систему, может управлять правами доступа к этой задаче.

Методика тестирования [1, 2].

Для каждой задачи Z формируется набор разбиений по стандартной схеме кросс-валидации «10×5-fold CV» и девять наборов по десять случайных разбиений с одинаковой длиной обучения (от 10% до 90% от длины выборки с шагом в 10%). В контроль и обучение объекты разных классов попадают в такой же пропорции, как и в исходной задаче. Полученные разбиения сохраняются в базе данных системы. Если задача Z однажды тестировалась в Полигоне, то система всегда будет использовать разбиения, сформированные при первом тестировании задачи. Тем самым гарантируется, что все алгоритмы тестируются на одинаковых исходных данных.

Репозиторий алгоритмов классификации.

Алгоритм классификации A принимает на входе:

— матрицу обучающей выборки $F = (F_{ij}) \in \mathbb{R}^{\ell\times n}$,
— вектор $y = (y_i)_{i=1}^{\ell} \in Y^\ell$ правильных ответов на обучающей выборке,
— матрицу $C = (C_{uv}) \in \mathbb{R}^{m\times m}$ стоимости потерь,
— вектор информации I о типах признаков,
— вектор W мета-параметров алгоритма,
— матрицу тестовой выборки $F' = (F'_{ij}) \in \mathbb{R}^{k\times n}$,

и выдаёт на выходе:

— ответы $y' = (y'_i) \in Y^k$ на тестовой выборке,
— матрицу оценок $P' = (P'_{iv}) \in [0,1]^{k\times m}$ принадлежности i-го объекта тестовой выборки v-му классу задачи.

Мета-параметры W задаются пользователем перед запуском алгоритма на тестирование. Экземпляры алгоритма A с различными значениями мета-параметров W (а также различных версий) считаются различными, и результаты их тестирования сохраняются в отдельных записях. Это позволяет Полигону лучше обеспечивать воспроизводимость по сравнению с TunedIt.

Взаимодействие с алгоритмами происходит при помощи специальной прослойки AlgProxy[7], которую должен реализовать автор алгоритма. AlgProxy обеспечивает обмен данными между алгоритмом A и веб-сервисом Полигона. Модель вза-

[6] http://weka.wikispaces.com/ARFF+(stable+version)

[7] В первых версиях Полигона [1, 2] взаимодействие с алгоритмами строилось на манер TunedIt при помощи локальной программы «Вычисляющий сервер». В дальнейшем была выбрана модель взаимодействия при помощи веб-сервиса, и необходимость в «Вычисляющем сервере» отпала.

имодействия основана на подходе «клиент-сервер», где AlgProxy — это клиент, а Полигон — это сервер. Инициатором взаимодействия выступает AlgProxy, который постоянно опрашивает Полигон «не появились ли задания для алгоритма A?». Если ответ положительный, то AlgProxy получает задание и начинает выполнение алгоритма A. Задание содержит данные задачи Z, параметры алгоритма W и набор разбиений $S = (S_q)$. AlgProxy тестирует алгоритм A на разбиениях S и передает результаты в систему Полигон. В упрощенном виде схема работы AlgProxy выглядит так:

1) запросить новое задание;
2) если задания нет, то через N секунд перейти к первому пункту;
3) если задание есть, то запросить данные задачи классификации;
4) провести тестирование алгоритма;
5) отправить результаты тестирования на сервер Полигона.

В документации[8] приведена пошаговая инструкция реализации AlgProxy для подключения собственного алгоритма.

Оценки качества алгоритма для задачи.

Статистика — это функция, которая принимает на входе:

— данные задачи классификации Z,
— набор разбиений $S = (S_q)$ задачи Z, использованных при тестировании алгоритма,
— результаты работы алгоритма (y'_q) и (P'_q) на каждом q-ом разбиении,

и выдаёт на выходе:

— скалярное или векторное значение, описывающее одну из характеристик качества алгоритма.

В текущей версии Полигона рассчитываются следующие статистики (некоторые из них подробнее обсуждались в [2]):

— частота ошибок на обучении и на контроле, а также средняя переобученность (с доверительными интервалами);
— среднее смещение и средняя вариация (характеристики адекватности и устойчивости модели);
— доля «шумовых» (где алгоритм часто ошибается), «пограничных» и «эталонных» (на которых алгоритм ошибается редко) объектов;
— распределение частоты ошибок и переобученности (показатели устойчивости классификации);
— зависимость переобученности и частоты ошибок от длины обучения;
— карта ошибок — точечный график: по оси X частота ошибок на обучении, по оси Y частота ошибок на контроле, каждая точка на карте соответствует одному разбиению;
— разделение ошибок на смещение и вариацию (анализ качества модели [3, 2]);
— ROC-кривая и площадь под ROC-кривой (анализ соотношения ошибок I и II рода [5]);
— распределение отступов (margin) объектов: по оси X — объекты, по оси Y — средний отступ объекта от границы класса (ещё одно разделение объектов на «шумовые», «пограничные» и «эталонные»).

Большинство статистик вычисляются как по всей выборке, так и отдельно по классам, а также отдельно для обучающей и контрольной выборки (если это осмысленно).

Результаты расчетов могут быть выгружены в «сыром» виде, что позволяет пользователю дополнить имеющиеся статистики собственными.

Пока текущая версия Полигона имеет два существенных ограничения: поддерживается только один тип задач (классификации) и отсутствует интерфейс и документация на английском языке.

Выводы

Благодаря архитектурным решениям, основанным на web-сервисах, и стандартизованной методике тестирования, система Полигон обладает лучшими характеристиками воспроизводимости по сравнению с системами TunedIt и MLComp. Полигон не накладывает ограничений на язык программирования и операционную систему при реализации алгоритмов. Алгоритмы исполняются на локальных машинах, что делает возможным тестирование их коммерческих версий. Применяемая методика тестирования позволяет использовать Полигон не только как средство вычисления и хранения результатов тестирования, но и как инструмент для исследования задач и алгоритмов классификации.

Литература

[1] *Воронцов К. В., Инякин А. С., Лисица А. В.* Система эмпирического измерения качества алгоритмов классификации // Всеросс. конф. ММРО-13. — М.: МАКС Пресс, 2007. — С. 577–581.

[2] *Воронцов К. В., Ивахненко А. А, Инякин А. С., Лисица А. В., Минаев П. Ю.* «Полигон» — распределённая система для эмпирического анализа задач и алгоритмов классификации // Всеросс. конф. ММРО-14. — М.: МАКС Пресс, 2009. — С. 503–506.

[3] *Domingos P.* A unified bias-variance decomposition and its applications: ICML'17. — 2000. — Pp. 231–238.

[4] *Frank A., Asuncion A.* UCI Machine Learning Repository // University of California, Irvine, School of Information and Computer Sciences, 2010. `www.ics.uci.edu/~mlearn/MLRepository.html`.

[5] *Hand D., Till R.* A simple generalization of area under the ROC curve for multiple class classification problems // Machine Learning, 45. — 2001. — Pp. 171–186.

[8] Документация системы в формате *wiki* расположена по адресу `http://www.MachineLearning.ru/wiki/index.php?title=Полигон_алгоритмов/Документация`

Точные оценки вероятности переобучения для несимметричных многомерных семейств алгоритмов*

Ботов П. В.
pbotov@forecsys.ru
Долгопрудный, Московский физико-технический институт (государственный университет)

Данная статья посвящена развитию методов модельных семейств алгоритмов, наиболее приближенных к реальным семействам, с дальнейшей целью получения по ним точных оценок вероятности переобучения. Рассмотрены несимметричные многомерные монотонная сетка, связка цепочек и унимодальная сетка. Для всех трёх семейств алгоритмов выведены теоремы о вероятности переобучения.

Exact bounds on probability of overfitting for asymmetric multidimensional sets of algorithms*

Botov P. V.
Moscow Institute of Physics and Technology (State University), Dolgoprudny, Russia

This article is devoted to the development of methods of model families of algorithms, closest to the real families, with a further purpose of obtaining accurate estimates of the probability of overfitting. We consider asymmetric multidimensional monotone set, chain pencil and unimodal set. For all three families of algorithms are derived theorems on probability of overfitting.

Введение

Комбинаторная теория обобщающей способности в настоящее время позволяет получать точные (не асимптотические и не завышенные) оценки вероятности переобучения, главным образом, для модельных семейств алгоритмов. В [5] такие оценки были получены для монотонных и унимодальных цепочек алгоритмов, единичной окрестности лучшего алгоритма, слоёв и интервалов булева куба, пары алгоритмов; в [3] — для связки монотонных цепочек; в [2] — для хэммингова шара и некоторых его подмножеств, в [1] — для многомерных монотонных и унимодальных сеток алгоритмов.

Модельные семейства вряд ли могут порождаться практическими задачами. Тем не менее, они интересны тем, что обладают свойствами расслоения и связности, благодаря которым переобучение существенно понижается, и которыми с необходимостью должны обладать реальные семейства, чтобы их можно было применять на практике [4].

Некоторые модельные семейства (сетки и связки) обладают также свойством «размерности» или «числа степеней свободы», что ещё больше сближает их с реальными семействами. В [1] было показано, что вероятность переобучения нейронных сетей, решающих деревьев, байесовских классификаторов на реальных задачах классификации неплохо приближается вероятностью переобучения многомерной монотонной сетки при соответствующем подборе её размерности.

Работа поддержана РФФИ (проект №08-07-00422) и программой ОМН РАН «Алгебраические и комбинаторные методы математической кибернетики и информационные системы нового поколения».

Сетки и связки, рассматривавшиеся ранее, являлись симметричными в том смысле, что все размерности давали равный вклад в оценку вероятности переобучения. В данной работе предлагается обобщение этих многомерных модельных семейств. Вводится ограничение на максимальную высоту D_i по каждой i-й размерности. Интуитивно, этот параметр показывает, насколько «богата» данная размерность; точнее — сколько алгоритмов, различимых на генеральной совокупности, можно получить, варьируя вектор параметров семейства вдоль i-й размерности. Такое обобщение ещё больше приближает модельные семейства к реальным.

Основные понятия и обозначения

Задано множество объектов $\mathbb{X} = \{x_1, \ldots, x_L\}$, называемое *генеральной выборкой*, конечное множество A, элементы которого называются *алгоритмами*, и бинарная функция ошибки $I\colon A \times \mathbb{X} \to \to \{0, 1\}$. Если $I(a, x) = 1$, то алгоритм a ошибается на объекте x, иначе a не ошибается на x.

Вектором ошибок алгоритма a называется бинарный вектор $\big(I(a, x_i)\big)_{i=1}^{L}$.

Число ошибок алгоритма a на выборке $X \subseteq \mathbb{X}$ есть $n(a, X) = \sum\limits_{x \in X} I(a, x)$.

Частота ошибок алгоритма a на выборке $X \subseteq \mathbb{X}$ есть $\nu(a, X) = \frac{1}{|X|} n(a, X)$.

Подмножество $A_t = \{a \in A\colon n(a, \mathbb{X}) = t + m\}$ называется *t-слоем* множества алгоритмов A относительно выборки $\mathbb{X}$, где m — число ошибок *лучшего алгоритма*, $m = \min\limits_{a \in A} n(a, \mathbb{X})$.

Методом обучения называется отображение $\mu\colon 2^{\mathbb{X}} \to A$, которое произвольной обучающей выборке X ставит в соответствие некоторый алгоритм $a = \mu X$ из A. Метод обучения μ называ-

ется методом минимизации эмпирического риска (МЭР), если для любой обучающей выборки X

$$\mu X \in A(X), \quad A(X) = \operatorname{Arg}\min_{a\in A} n(a, X).$$

Если множество $A(X)$ содержит более одного алгоритма, то будем полагать, что реализуется худший случай. Метод МЭР называется *пессимистическим* (ПМЭР), если

$$\mu X \in \operatorname{Arg}\max_{a\in A(X)} n(a, \mathbb{X}).$$

Допустим, что все C_L^ℓ разбиений генеральной выборки $\mathbb{X} = X \sqcup \bar{X}$ на наблюдаемую обучающую выборку X длины ℓ и скрытую контрольную выборку $\bar{X}$ длины $k = L - \ell$ равновероятны.

Нашей основной задачей будет вычисление вероятности переобучения для ПМЭР:

$$Q_\varepsilon = \frac{1}{C_L^\ell} \sum_{X\sqcup\bar{X}} \big[\nu(\mu X, \bar{X}) - \nu(\mu X, X) \geqslant \varepsilon\big].$$

Вектор $\boldsymbol{d} = (d_i)_{i=1}^h$ с целыми неотрицательными компонентами будем называть *неотрицательным вектором индексов*. Положим $|\boldsymbol{d}| = \sum\limits_{i=1}^h |d_i|$. На множестве таких векторов введём частичный порядок: $\boldsymbol{d} < \boldsymbol{d}'$ тогда и только тогда, когда $d_i \leqslant d'_i$ для всех $i \in 1, \ldots, h$, и хотя бы одно из неравенств строгое.

Базовые гипотеза и леммы

Согласно гипотезе [5] о порождающих и запрещающих множествах условие $[\mu X = a]$ выражается через сумму элементов вида $[X_{av} \in X][X'_{av} \in \bar{X}]$. В настоящей работе вводится гипотеза, более удобная в случае многомерных семейств.

Гипотеза 1. Для каждого t-слоя A_t можно указать такое множество индексов V_t, и для каждого индекса $v \in V_t$ *порождающее* множество $X_{tv} \subset \mathbb{X}$, непересекающееся с ним *запрещающее* множество $X'_{tv} \subset \mathbb{X}$ и коэффициент $c_{tv} \in \mathbb{Z}$, такие, что

$$[\mu X \in A_t] = \sum_{v\in V_t} c_{tv}[X_{tv} \subseteq X][X'_{tv} \subseteq \bar{X}], \quad (1)$$

причём $n(a, X_{tv}) = 0$ и $n(a, X'_{tv}) = t$ для всех $a \in A_t$.

Лемма 1. *Если гипотеза 1 верна, то вероятность переобучения есть $Q_\varepsilon = \sum_{t=0}^k Q_{\varepsilon,t}$, где $Q_{\varepsilon,t}$ — вклад t-слоя в вероятность переобучения:*

$$Q_{\varepsilon,t} = \sum_{v\in V_t} c_{tv} \frac{C_{L_{tv}}^{\ell_{tv}}}{C_L^\ell} H_{L_{tv}}^{\ell_{tv},m}(s_t(\varepsilon)),$$

где $L_{tv} = L - |X_{tv}| - |X'_{tv}|$, $\ell_{tv} = \ell - |X_{tv}|$, $s_t(\varepsilon) = \frac{\ell}{L}(m + t - \varepsilon k)$, $H_L^{\ell,m}(z)$ — функция гипергеометрического распределения:

$$H_L^{\ell,m}(z) = \sum_{s=s_0}^{\lfloor z\rfloor} \frac{C_m^s C_{L-m}^{\ell-s}}{C_L^\ell}, \quad s_0 = \max\{0, m-k\}.$$

Монотонная сетка алгоритмов

Определение 1. *Множество алгоритмов $A = \{a_{\boldsymbol{d}} \colon \boldsymbol{d} \leqslant \boldsymbol{D},\ |\boldsymbol{d}| < D_0\}$, задаваемое неотрицательным вектором высот $\boldsymbol{D}$ и общей высотой D_0, называется монотонной h-мерной сеткой алгоритмов, если выполнены условия:*

1) *если $\boldsymbol{d} < \boldsymbol{d}'$, то для всех $x_i \in \mathbb{X}$ выполнено $I(a_{\boldsymbol{d}}, x_i) \leqslant I(a_{\boldsymbol{d}'}, x_i)$, причём ровно $|d - d'|$ неравенств строгие;*
2) *$n(a_{\boldsymbol{d}}, \mathbb{X}) = m + |\boldsymbol{d}|$ при некотором $m \geqslant 0$ для всех $a_{\boldsymbol{d}} \in A$. Алгоритм $a_{\boldsymbol{0}}$ называется лучшим.*

Из определения следует, что все объекты $x \in \mathbb{X}$ можно отнести к одной из трёх категорий:

1) *шумовые объекты*, на которых любой алгоритм из A ошибается; их ровно m;
2) *безошибочные*, на которых ни один из алгоритмов из A не ошибается;
3) все остальные объекты x_i^j, где $j = 1, \ldots, h$, $i = 1, \ldots, \min(D_j, D_0)$, причём $I(a_{\boldsymbol{d}}, x_i^j) = [d_j \geqslant i]$.

Монотонная сетка — это модель семейства алгоритмов с h непрерывными параметрами, в которой по мере увеличения j-го параметра ошибки возникают последовательно на объектах $x_1^j, \ldots, x_{D_j}^j$.

Пример 1. Монотонная двумерная сетка для выборки из $L = 5$ объектов, с $m = 0$ и $\boldsymbol{D} = (2, 3)$:

	a_{00}	a_{10}	a_{20}	a_{01}	a_{11}	a_{21}	a_{02}	a_{12}	a_{22}	a_{03}	a_{13}	a_{23}
x_1^1	0	**1**	1	0	**1**	1	0	**1**	1	0	**1**	1
x_2^1	0	0	**1**	0	0	**1**	0	0	**1**	0	0	**1**
x_1^2	0	0	0	**1**	**1**	**1**	1	1	1	1	1	1
x_2^2	0	0	0	0	0	0	**1**	**1**	**1**	1	1	1
x_3^2	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**

Теорема 2 (О монотонной h-мерной сетке). *Пусть A — h-мерная монотонная сетка, метод μ является ПМЭР, выполнено условие $|\boldsymbol{D}| \leqslant L - m$. Тогда вероятность переобучения $Q_\varepsilon = \sum\limits_{t=0}^{t_{\max}} Q_{\varepsilon,t}$, где $t_{\max} = \min(k, D)$ и $Q_{\varepsilon,t}$ — вклад t-слоя.*

В случае $t < t_{\max}$

$$Q_{\varepsilon,t} = \sum_{j=0}^h \alpha(\boldsymbol{D}, t, j) \frac{C_{L-t-j}^{\ell-j}}{C_L^\ell} H_{L-t-j}^{\ell-j,m}(s_t(\varepsilon)),$$

в случае $t = t_{\max}$

$$Q_{\varepsilon,t} = 1 - \sum_{i=0}^t \sum_{j=0}^h \alpha(\boldsymbol{D}, i, j) \frac{C_{L-i-j}^{\ell-j}}{C_L^\ell} H_{L-i-j}^{\ell-j,m}(s_t(\varepsilon)),$$

где числовые параметры α вычисляются через

$$\alpha(\boldsymbol{D}, t, j) = \sum_{|\boldsymbol{d}|=t} \sum_{\substack{|I|=j \\ I\subseteq E}} \Big(\prod_{q\in I} [d_q < D_q]\Big)\Big(\prod_{r\in\bar{I}} [d_r = D_r]\Big),$$

где $E = \{1, \ldots, h\}$ и $\bar{I} = E \setminus I$.

Связка монотонных цепочек

Определение 2. *Множество алгоритмов* $A = \{a_d^b\colon b = 1,\dots,h,\ d = 0,\dots,D_b\}$, *называется связкой* h *монотонных цепочек алгоритмов высоты* $\boldsymbol{D} = (D_1,\dots,D_h)$, *если*

1) *существует алгоритм* $a^* \in A$, *называемый лучшим в связке, такой, что* $a^* = a_0^b$ *для всех* b;
2) $n(a_d^b, \mathbb{X}) = m + d$ *при некотором* $m \geqslant 0$, *любом* $b \in \{1,\dots,h\}$ *и любом* $d \leqslant D_b$;
3) *подмножество* $A_b = \{a_d^b : d = 0,\dots,D_b\}$ *является монотонной цепочкой для всех* $b \in \{1,\dots,h\}$;
4) *если для некоторого объекта* x_i *выполняется* $I(a_d^b, x_i) = I(a_{d'}^{b'}, x_i) = 1,\ b \neq b'$, *то* $I(a^*, x_i) = 1$.

Из определения следует, что все объекты $x \in \mathbb{X}$ можно отнести к одной из трёх категорий:

1) *шумовые объекты*, на которых любой алгоритм из A ошибается; их ровно m;
2) *безошибочные объекты*, на которых ни один из алгоритмов из A не ошибается;
3) все остальные объекты x_i^j, где $j = 1,\dots,h$, $i = 1,\dots,D_j$, причём $I(a_d^b, x_i^j) = [d \geqslant i][j = b]$.

Пример 2. Связка из трёх цепочек с $L = 8$, $m = 1$ и $\boldsymbol{D} = (1,2,3)$:

	a^*	a_1^1	a_1^2	a_2^2	a_1^3	a_2^3	a_3^3
x_1^0	1	1	1	1	1	1	1
x_2^0	0	0	0	0	0	0	0
x_1^1	0	**1**	0	0	0	0	0
x_1^2	0	0	**1**	1	0	0	0
x_2^2	0	0	0	**1**	0	0	0
x_1^2	0	0	0	0	**1**	1	1
x_2^3	0	0	0	0	0	**1**	1
x_3^3	0	0	0	0	0	0	**1**

Теорема 3 (О связке h цепочек). *Пусть* A — *связка из* h *цепочек, метод* μ *является ПМЭР и выполнены условия* $|\boldsymbol{D}| + m \leqslant L,\ D_i \leqslant k,\ i \in 1,\dots,h$. *Тогда вероятность переобучения есть* $Q_\varepsilon = \sum\limits_{t=0}^{k} Q_{\varepsilon,t}$, *где* $Q_{\varepsilon,t}$ — *вклад* t*-слоя, который равен*

$$Q_{\varepsilon,t} = \sum_{i=1}^{h} \sum_{|I|=i} \omega_2 \sum_{\theta=0}^{(t-1)|\bar{I}|} \sum_{r=0}^{|\bar{I}|} \alpha(t,\theta,r,\boldsymbol{D},\bar{I}) \times \times \frac{C_{\ell-r-\omega_1}^{L-r-\omega_1-\theta-t|I|}}{C_L^\ell} H_{\ell-r-\omega_1,m}^{L-r-\omega_1-\theta-t|I|}(s_t(\varepsilon)),$$

где $I \subseteq E,\ \bar{I} = E\backslash I,\ \omega_1 = \sum\limits_{p\in I}[t < D_p],\ \omega_2 = \prod\limits_{p\in I}[t \leqslant D_p]$;

$$\alpha(t,\theta,r,\boldsymbol{D},\bar{I}) = \sum_{|\boldsymbol{d}|=\theta} \sum_{|J|=r} \Big(\prod_{p=1}^{|J|}[d_p < D_{J_p}][d_p < t]\Big) \times \times \Big(\prod_{q=1}^{|\bar{J}|}[d_q = D_{\bar{J}_q}][d_q < t]\Big),$$

$J \subseteq \bar{I}$, $\bar{J} = \bar{I}/J$, J_p *и* $\bar{J}_q$ — p*-ый и* q*-ый элемент подмножеств* J *и* $\bar{J}$ *соответственно, а* $\boldsymbol{d}$ — *неотрицательный вектор индексов размерности* $|\bar{I}|$.

Унимодальная сетка

Аналогично неотрицательному вектору индексов $\boldsymbol{d}$, введём вектор индексов $\boldsymbol{w} = (w_i)_{i=1}^h$, с которого требование неотрицательности снято.

Определение 3. *Множество алгоритмов* $A = \{a_{\boldsymbol{w}}\colon -\boldsymbol{W}^- \leqslant \boldsymbol{w} \leqslant \boldsymbol{W}^+,\ |\boldsymbol{w}| \leqslant W_0\}$, *задаваемое двумя векторами высот* $-\boldsymbol{W}^- \leqslant \boldsymbol{0} \leqslant \boldsymbol{W}^+$ *и общей высотой* W_0, *называется унимодальной* h*-мерной сеткой алгоритмов, если выполнены условия:*

1. *если* $\boldsymbol{0} \leqslant \boldsymbol{w} < \boldsymbol{w}'$ *или* $\boldsymbol{w}' < \boldsymbol{w} \leqslant \boldsymbol{0}$, *то для всех* $x_i \in \mathbb{X}$ *выполнено* $I(a_{\boldsymbol{w}}, x_i) \leqslant I(a_{\boldsymbol{w}'}, x_i)$, *причём ровно* $|w - w'|$ *неравенств строгие.*
2. $n(a_{\boldsymbol{w}}, \mathbb{X}) = m + |\boldsymbol{w}|$ *при некотором* $m \geqslant 0$ *для всех* $a_{\boldsymbol{w}} \in A$. *Алгоритм* $a_{\boldsymbol{0}}$ *называется лучшим.*

Из определения следует, что все объекты $x \in \mathbb{X}$ можно отнести к одной из трёх категорий:

1) *шумовые объекты*, на которых любой алгоритм из A ошибается; их ровно m;
2) *безошибочные объекты*, на которых ни один из алгоритмов из A не ошибается;
3) все остальные объекты x_i^j, где $j = 1,\dots,h$, $i = -W_j^-,\dots,W_j^+$, и $I(a_{\boldsymbol{w}}, x_i^j) = [w_j \geqslant i > 0] + + [w_j \leqslant i < 0]$.

Унимодальная сетка — это модель семейства алгоритмов с h непрерывными параметрами, в которой по мере увеличения j-го параметра ошибки возникают последовательно на объектах $x_1^j,\dots,x_{W_j^+}^j$, а при уменьшении — на $x_{-1}^j,\dots,x_{-W_j^-}^j$.

Теорема 4 (Об унимодальной h-мерной сетке). *Пусть* A — h*-мерная унимодальная сетка, метод* μ *является ПМЭР и выполнены условия* $|\boldsymbol{W}^-| + |\boldsymbol{W}^+| \leqslant L - m$ *и* $W_0 \leqslant k$. *Тогда вероятность переобучения есть* $Q_\varepsilon = \sum\limits_{t=0}^{W_0} Q_{\varepsilon,t}$, *где* $Q_{\varepsilon,t}$ — *вклад* t*-слоя, который при* $t < W_0$ *равен*

$$Q_{\varepsilon,t} = \sum_{|\boldsymbol{d}|=t} \sum_{\theta=0}^{h} \sum_{\substack{I\subseteq E\\|I|=\theta}} \sum_{q=0}^{|I|} \sum_{\substack{J\subseteq I\\|J|=q}} \delta_1(\bar{J},\boldsymbol{d},\boldsymbol{W}^\pm) \times \times \sum_{p=0}^{q} \sum_{\substack{M\subseteq J\\|M|=p}} \delta_2(M,\bar{M},\boldsymbol{d},\boldsymbol{W}^\pm) \sum_{r=0}^{|\bar{I}|} \delta_3(r,\bar{I},\boldsymbol{d},\boldsymbol{W}^\pm) \times \times \frac{C_{L-q-p-r-t-\eta(\boldsymbol{d},I)}^{\ell-q-p-r}}{C_L^\ell} H_{L-q-p-r-t-\eta(\boldsymbol{d},I)}^{\ell-q-p-r,m}(s_t(\varepsilon)),$$

где $E = \{1,\dots,h\}$, $\bar{I} = E\backslash I$, $\bar{J} = I\backslash J$, $\bar{M} = J\backslash M$, $\eta(\boldsymbol{d},I) = \sum\limits_{i\in I} d_i$, *и целочисленные коэффициенты* δ

вычисляются следующим образом:

$$\delta_1(\bar{J}, \boldsymbol{d}, \boldsymbol{W}^{\pm}) = \prod_{i\in\bar{J}}\Big([d_i = W_i^+][d_i = W_i^-] - [d_i > 0][d_i = W_i^+][d_i \leqslant W_i^-] - [d_i > 0][d_i \leqslant W_i^+][d_i = W_i^+]\Big);$$

$$\delta_2(M, \bar{M}, \boldsymbol{d}, \boldsymbol{D}^{\pm}) = \prod_{j\in M}[d_j < W_j^+][d_j < W_j^-] \times \prod_{i\in\bar{M}}\Big([d_i = W_i^+][d_i < W_i^-] + [d_i < W_i^+][d_i = W_i^-] - [d_i > 0][d_i < W_i^+][d_i \leqslant W_i^-] - [d_i > 0][d_i \leqslant W_i^+][d_i < W_i^-]\Big);$$

$$\delta_3(r, \bar{I}, \boldsymbol{d}, \boldsymbol{D}^{\pm}) = \sum_{\substack{T\subseteq\bar{I}\\|T|=r}}\prod_{i\in T}[d_i > 0]\Big([d_i < W_i^+] + [d_i < W_i^-]\Big) \times \prod_{j\in\bar{I}\setminus T}[d_j > 0]\Big([d_j = W_j^+] + [d_j = W_j^-]\Big).$$

Эксперимент

В силу ограничения объёма статьи приведём результаты лишь одного эксперимента. Построим монотонную сетку размерности h с общей высотой D_0 и вектором высот $\boldsymbol{D} = (D_0, \ldots, D_0, D_*)$. Будем варьировать высоту последней размерности D_* от 0 до D_0. Графики зависимости вероятности переобучения Q_ε от D_* при различных h показаны на рис. 1.

Судя по графикам, основной рост Q_ε происходит при $D_* \leqslant 6$, а далее вероятность переобучения практически не меняется. Это согласуется с выводами предыдущих работ о том, что вероятность переобучения определяется нижними слоями семейства. Отсюда также следует и качественно новый вывод: добавление даже «не слишком богатой» размерности (высоты порядка 6) может приводить к значительному росту вероятности переобучения.

Выводы

Для трёх модельных многомерных семейств алгоритмов — монотонной сетки, связки монотонных цепочек и унимодальной сетки, получены точные оценки вероятности переобучения для случая, когда размерности имеют различную высоту. Если все высоты одинаковы, эти оценки совпадают с полученными ранее в [1].

Высоту размерности можно интерпретировать как «меру сложности» данной размерности. Она показывает, сколько алгоритмов, различимых на генеральной совокупности, можно получить, варьируя вектор параметров семейства вдоль данной

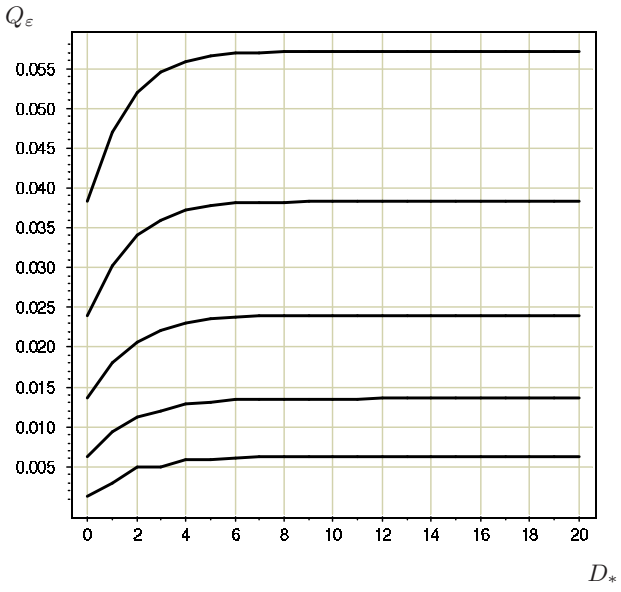

Рис. 1. Зависимость вероятности переобучения Q_ε монотонных сеток с общей высотой D_0 и вектором высот $\boldsymbol{D} = (D_0, \ldots, D_0, D_*)$ от параметра D_* при изменении h от 1 (нижний график) до 5 (верхний график), при $L = 500$, $\ell = 300$, $m = D_0 = 20$, $\varepsilon = 0{,}05$.

размерности. Другая интерпретация — сколько допустимых значений может принимать данная размерность.

Вычисления показали, что с ростом высоты одной из размерностей вероятность переобучения быстро возрастает и затем стабилизируется. Это означает, что «сложности» отдельных размерностей не так важны для определения вероятности переобучения, как их количество h.

Литература

[1] *Ботов П. В.* Точные оценки вероятности переобучения для монотонных и унимодальных семейств алгоритмов // Всеросс. конф. ММРО-14. — М.: МАКС Пресс, 2009. — С. 7–10.

[2] *Толстихин И. О.* Вероятность переобучения плотных и разреженных семейств алгоритмов // Межд. конф. Интеллектуализация обработки информации ИОИ-8, 2010. — С. 27–30 (в этом же сборнике).

[3] *Frey A. I.* Accurate estimates of the generalization ability for symmetric sets of predictors and randomized learning algorithms // Pattern Recognition and Image Analysis. — 2010. — Vol. 20, No. 3. — Pp. 241–250.

[4] *Vorontsov K. V.* Splitting and similarity phenomena in the sets of classifiers and their effect on the probability of overfitting // Pattern Recognition and Image Analysis. — 2009. — Vol. 19, No. 3. — Pp. 412–420.

[5] *Vorontsov K. V.* Exact combinatorial bounds on the probability of overfitting for empirical risk minimization // Pattern Recognition and Image Analysis. — 2010. — Vol. 20, No. 3. — Pp. 269–285.

Вероятность переобучения плотных и разреженных семейств алгоритмов*

Толстихин И. О.
iliya.tolstikhin@gmail.com
Вычислительный центр им. А. А. Дородницына РАН

Показано, что для оценивания вероятности переобучения семейств алгоритмов, обладающих свойствами расслоения и связности, достаточно брать их подмножества, состоящие из небольшого числа существенно различных алгоритмов с малым числом ошибок.

The probability of overfitting for the compact and sparse sets of predictors*

Tolstikhin I. O.
Dorodnicyn Computing Centre of RAS, Moscow, Russia

We show that if a set of classifiers possesses the properties of splitting and similarity, then its probability of overfitting can be approximated by a small subset of essentially different classifiers with low error rate.

Оценивание вероятности переобучения — одна из основных задач теории статистического обучения [2]. Чем точнее её решение, тем больше возможностей открывается для создания алгоритмов машинного обучения с гарантированно высокой способностью к обобщению эмпирических данных.

Большинство известных верхних оценок вероятности переобучения представляют собой произведение двух сомножителей. Первый описывает сложность семейства алгоритмов. Например, в VC-оценках [1] это *коэффициент разнообразия* (shattering coefficient), равный числу алгоритмов семейства, различимых на заданной конечной выборке длины L. Второй сомножитель — это вероятность большого уклонения частот ошибок в двух непересекающихся подвыборках длины $L/2$ для одного отдельно взятого алгоритма. Обычно она оценивается сверху неравенствами Хёффдинга, Чернова, или другими оценками скорости сходимости в законе больших чисел или его обобщениях [2].

В комбинаторном подходе [5, 6] рассматривается *генеральная выборка* длины L и предполагается равновероятность всех её разбиений на две подвыборки — наблюдаемую *обучающую* длины ℓ и скрытую *контрольную* длины $k = L - \ell$. В этом случае второй сомножитель определяется точно, через гипергеометрическое распределение.

Первый сомножитель сильно завышен в большинстве известных оценок. Степень его завышенности измерялась в экспериментах на реальных задачах классификации [4]. Для этого вероятность переобучения оценивалась методом скользящего контроля. Поделив её на второй сомножитель, который известен точно, можно получить *эффективный локальный коэффициент разнообразия* (ЭЛКР). Он показывает, каким должен был бы быть первый сомножитель, чтобы оценка не была завышенной. Он также интерпретируется как число алгоритмов, «эффективно используемых» при решении данной конкретной задачи, то есть имеющих высокую вероятность быть выбранными в результате обучения. В экспериментах [4] ЭЛКР принимал значения порядка 10^0–10^1 при коэффициентах разнообразия порядка 10^6–10^9. Ни одна из известных теорий не могла объяснить столь низких значений ЭЛКР. В данной работе делается попытка дать такое объяснение.

В последующих работах [5, 6] было показано, что завышенность VC-оценок возможно устранить только путём одновременного учёта двух эффектов, наблюдаемых при решении реальных задач.

Эффект расслоения связан с тем, что при фиксации задачи семейство алгоритмов расслаивается по числу ошибок на генеральной выборке. Чем выше слой, тем меньше вероятность, что в результате обучения будет выбран алгоритм из этого слоя. Как правило, в нижних слоях находится ничтожно малая доля алгоритмов, однако именно они «эффективно используются» в данной задаче.

Эффект связности возникает в семействах алгоритмов, непрерывных по параметрам. В этом случае для каждого алгоритма в семействе найдётся некоторое число алгоритмов, различимых с ним только на одном каком-то объекте генеральной выборки. Будем называть такие алгоритмы связанными. Похожие (в частности, связанные) алгоритмы вносят меньший вклад в вероятность переобучения, чем непохожие. Поэтому увеличение связности способствует снижению переобучения.

Понимание этих эффектов позволяет выдвинуть следующую гипотезу: *вероятность переобучения расслоенного и связного множества алго-*

Работа поддержана РФФИ (проект №08-07-00422) и программой ОМН РАН «Алгебраические и комбинаторные методы математической кибернетики и информационные системы нового поколения».

ритмов может быть аппроксимирована вероятностью переобучения его подмножества, состоящего из существенно различных алгоритмов нижних слоёв. При этом малые значения ЭЛКР, наблюдавшиеся в экспериментах, позволяют надеяться, что для аппроксимации хватит нескольких десятков алгоритмов. В данной работе исследуется возможность аппроксимации «плотных» семейств алгоритмов их «разреженными» подсемействами.

Понятие вероятности переобучения

Задано множество объектов $\mathbb{X} = \{x_1, \ldots, x_L\}$, множество алгоритмов $A = \{a_1, \ldots a_D\}$ и бинарная функция $I\colon \mathbb{X} \times A \to \{0, 1\}$, называемая *индикатором ошибки*, такая, что $I(a, x) = 1$ тогда и только тогда, когда алгоритм a ошибается на объекте x.

Каждому алгоритму соответствует бинарный *вектор ошибок* $\big(I(a, x_i)\big)_{i=1}^{L}$ длины L. *Матрицей ошибок* называется $D \times L$-матрица, строками которой являются векторы ошибок алгоритмов из A. В дальнейшем будем считать, что строки матрицы ошибок попарно различны, и отождествлять алгоритмы с их векторами ошибок.

Числом ошибок алгоритма a на выборке $X \subseteq \mathbb{X}$ называется величина $n(a, X) = \sum\limits_{x \in X} I(a, x)$.

Частотой ошибок (или *эмпирическим риском*) алгоритма a на выборке $X \subseteq \mathbb{X}$ называется величина $\nu(a, X) = \frac{1}{|X|} n(a, X)$.

Обозначим через $[\mathbb{X}]^\ell$ множество всех C_L^ℓ подмножеств $X \subseteq \mathbb{X}$ мощности ℓ.

Методом обучения называется отображение $\mu\colon [\mathbb{X}]^\ell \to A$, которое произвольной выборке X длины ℓ ставит в соответствие некоторый алгоритм.

Метод обучения μ называется методом *минимизации эмпирического риска* (МЭР), если

$$\mu X \in A(X) \equiv \operatorname{Arg}\min_{a \in A} n(a, X).$$

В случае $|A(X)| > 1$ выбор алгоритма из $A(X)$ неоднозначен. Метод МЭР называется *пессимистическим* (ПМЭР), если

$$\mu X \in \operatorname{Arg}\max_{a \in A(X)} n(a, \mathbb{X}\backslash X).$$

Метод МЭР называется *рандомизированным* (РМЭР), если он выбирает произвольный алгоритм из $A(X)$ случайно и равновероятно [3].

Величина $\delta_\mu(X) = \nu\left(\mu X, \mathbb{X}\backslash X\right) - \nu\left(\mu X, X\right)$ называется *переобученностью* метода μ на выборке X. Если $\delta_\mu(X) \geqslant \varepsilon$, где ε — фиксированный *порог переобучения*, то говорят, что *метод μ переобучен на выборке X*.

Следуя комбинаторному подходу [4, 5, 6], будем полагать, что при фиксированном ℓ все C_L^ℓ разбиений *генеральной выборки* $\mathbb{X}$ на наблюдаемую обучающую выборку X длины ℓ и скрытую контрольную $\bar{X} = \mathbb{X}\backslash X$ длины $k = L - \ell$ равновероятны.

Нашей основной задачей будет вычисление вероятности переобучения для ПМЭР:

$$Q_\varepsilon(A) = P\big[\delta_\mu(X) \geqslant \varepsilon\big] = \frac{1}{C_L^\ell} \sum_{X \in [\mathbb{X}]^\ell} \big[\delta_\mu(X) \geqslant \varepsilon\big],$$

или для РМЭР:

$$Q_\varepsilon(A) = \sum_{a \in A} Q_{\varepsilon,a}(A),$$

где величина $Q_{\varepsilon,a}(A)$ называется *вкладом алгоритма a* в вероятность переобучения:

$$Q_{\varepsilon,a}(A) = \frac{1}{C_L^\ell} \sum_{X \in [\mathbb{X}]^\ell} \frac{[a \in A(X)]}{|A(X)|} \big[\delta(a, X)) \geqslant \varepsilon\big].$$

Заметим, что ПМЭР не реализуем на практике, т. к. он «подглядывает» в скрытую выборку на этапе обучения. Теоретически он интересен тем, что даёт верхние оценки вероятности переобучения. Введение РМЭР упрощает вывод точных оценок вероятности переобучения для семейств алгоритмов, обладающих определённой симметрией [3].

Шар алгоритмов и его подмножества

Пусть $d(a, a')$ — расстояние Хэмминга между алгоритмами a и a' как L-мерными бинарными векторами. Введём множества алгоритмов:

$B_r(a_0) = \big\{a \in \{0,1\}^L\colon d(a_0, a) \leqslant r\big\}$ — *шар алгоритмов* с центром в $a_0 \in \{0,1\}^L$ и радиусом r;

$S_r(a_0) = \big\{a \in B_r(a_0)\colon d(a_0, a) = r\big\}$ — *сфера алгоритмов* с центром a_0 и радиусом r;

$A_m = \big\{a \in \{0,1\}^L\colon n(a, \mathbb{X}) = m\big\}$ — *m-й слой*;

$B_r^m(a_0) = B_r(a_0) \cap A_m$ — *m-й слой шара* $B_r(a_0)$;

$S_r^m(a_0) = S_r(a_0) \cap A_m$ — *m-й слой сферы* $S_r(a_0)$.

Шар алгоритмов интересен тем, что это «максимально плотное» множество алгоритмов, обладающее наибольшей связностью среди всех множеств такой же мощности. Следовательно, оценки вероятности переобучения, получаемые для шара или его слоёв, являются оценками «лучшего случая» и могут служить ориентировочными нижними оценками и для других множеств алгоритмов.

Далее приводятся точные оценки вероятности переобучения для шара, его слоёв и некоторых их разреженных подмножеств. Будет показано, что в m-м слое сферы можно взять совсем небольшое подмножество алгоритмов, для которого вероятность переобучения лишь немного отличается от вероятности переобучения соответствующего слоя шара, содержащего на несколько порядков большее число алгоритмов.

Обозначим через $H_L^{l,m}(z)$ гипергеометрическую функцию распределения:

$$H_L^{l,m}(z) = \sum_{s=0}^{\lfloor z \rfloor} h_L^{l,m}(s), \quad h_L^{l,m}(s) = \frac{C_m^s C_{L-m}^{\ell-s}}{C_L^\ell}.$$

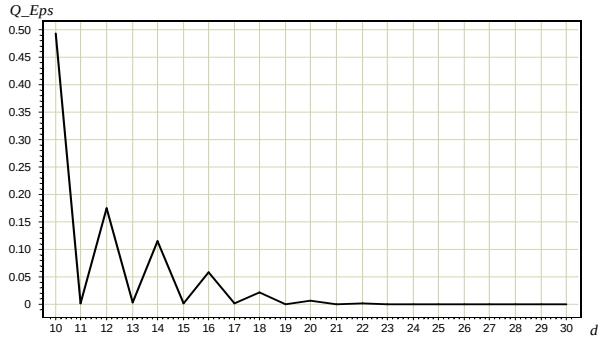

Рис. 1. Вклады слоёв шара в вероятность переобучения, при $l = k = 100,\ m = 20,\ r_0 = 10,\ \varepsilon = 0{,}05$.

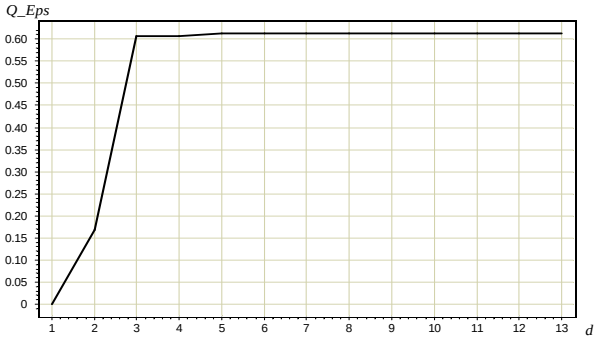

Рис. 2. Зависимость Q_ε от числа d нижних слоев шара, при $l = k = 100,\ m = 10,\ r_0 = 6,\ \varepsilon = 0,05$.

Теорема 1. *Пусть μ — РМЭР, $A = B_{r_0}(a_0)$ — шар алгоритмов, $n(a_0, \mathbb{X}) = m$ и $r_0 \leqslant \min(m, L-m)$. Тогда для любого $\varepsilon \in [0,1]$:*

$$Q_\varepsilon = \sum_{i=0}^{r_0} h_L^{\ell,m}(i)\, \frac{\sum\limits_{r=0}^{r_0}\sum\limits_{n=0}^{r} S(n,r,i)\big[m+r-2n \geqslant \varepsilon k\big]}{\sum\limits_{r=0}^{r_0}\sum\limits_{n=0}^{r} S(n,r,i)} + \sum_{i=r_0+1}^{\lfloor s_d(\varepsilon)\rfloor} h_L^{\ell,m}(i),$$

$$S(n,r,i) = C_{m-i}^{n-i} C_{k-m+i}^{r-n},\quad s_d(\varepsilon) = \tfrac{\ell}{L}(m-\varepsilon k) + \tfrac{r_0 k}{L}.$$

На рис. 1 показаны значения вкладов слоев шара в вероятность его переобучения. Видно, что наибольший вклад приходится на первые несколько слоев. Поэтому отдельный интерес представляет оценка вероятности переобучения для нескольких нижних слоев шара алгоритмов.

Теорема 2. *Пусть μ — РМЭР, $A = \big\{a \in B_{r_0}(a_0) : m - r_0 \leqslant n(a,\mathbb{X}) \leqslant m - r_0 + d - 1\big\}$ — d нижних слоёв шара, $n(a_0,\mathbb{X}) = m$ и $r_0 \leqslant \min(m, L-m)$. Тогда для любого $\varepsilon \in [0,1]$:*

$$Q_\varepsilon = \sum_{i=0}^{r_0} h_L^{\ell,m}(i)\, \frac{\sum\limits_{r=0}^{r_0}\sum\limits_{n=0}^{r} S'(n,r,i)\,\big[m+r-2n \geqslant \varepsilon k\big]}{\sum\limits_{r=0}^{r_0}\sum\limits_{n=0}^{r} S'(n,r,i)} + \sum_{i=r_0+1}^{\lfloor s_d(\varepsilon)\rfloor} h_L^{\ell,m}(i),$$

где $S'(n,r,i) = C_{m-i}^{n-i} C_{k-m+i}^{r-n}\,[r + r_0 + 1 \leqslant 2n + d]$.

Очевидно, что случай $d = 2r_0 + 1$ соответствует всему шару алгоритмов $B_{r_0}(a_0)$. На рис. 2 представлена зависимость вероятности переобучения нижних слоев шара от их числа. Видно, что вероятность переобучения шара достигается уже на 3–4 его нижних слоях.

Далее мы покажем возможность аппроксимации вероятности переобучения «плотного» семейства $B_{r_0}^m(a_0)$ его разреженным подмножеством.

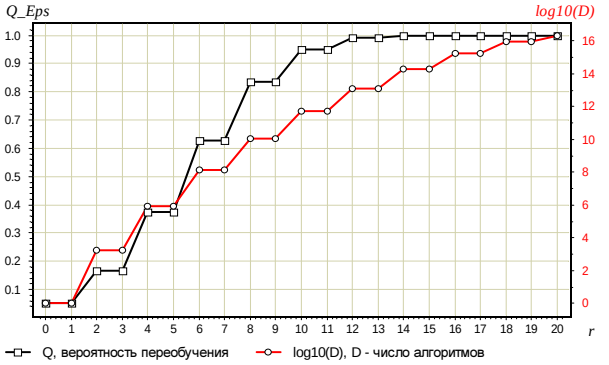

Рис. 3. Зависимость вероятности переобучения Q_ε и $\log_{10}|A|$ от радиуса шара r для m-го слоя шара, при $l = k = 100,\ m = 10,\ \varepsilon = 0{,}05$.

Теорема 3. *Пусть μ — МЭР, $A = B_{r_0}^m(a_0)$ — m-й слой шара, $m \leqslant k$ и $n(a_0,\mathbb{X}) = m$. Тогда для любого $\varepsilon \in [0,1]$:*

$$Q_\varepsilon = \begin{cases} H_L^{\ell,m}\big(\frac{\ell}{L}(m-\varepsilon k) + \lfloor r_0/2\rfloor\big), & m \geqslant \varepsilon k, \\ 0, & m < \varepsilon k. \end{cases}$$

На рис. 3 представлена зависимость вероятности переобучения и логарифма числа алгоритмов семейства $B_r^m(a_0)$ от радиуса r. Благодаря значительной «плотности» данного семейства вероятность переобучения остаётся на приемлемо низком уровне при мощности семейства порядка тысяч.

Следующая лемма показывает, что вероятность переобучения множества алгоритмов, лежащего в слое, может только увеличиться при добавлении к нему новых алгоритмов из того же слоя.

Лемма 4. *Пусть μ — МЭР, $A \subset A_m$ и $a \in A_m \setminus A$. Тогда $Q_\varepsilon(A) \leqslant Q_\varepsilon(A \cup \{a\})$.*

Зададимся целью найти в m-м слое шара подмножество B' малой мощности, чтобы их вероятности переобучения совпадали или хотя бы не сильно отличались: $Q_\varepsilon(B') \lessapprox Q_\varepsilon\big(B_{r_0}^m(a_0)\big)$. Следующая теорема ограничивает область поиска такого подмножества m-м слоем сферы $S_{2\lfloor r_0/2\rfloor}^m(a_0)$.

Введем обозначение $\delta = \lfloor r_0/2\rfloor$.

Теорема 5. *Пусть μ — МЭР, $m = n(a_0, \mathbb{X})$ и $k \geqslant m + \delta$. Тогда $Q_\varepsilon\big(B^m_{r_0}(a_0)\big) = Q_\varepsilon\big(S^m_{2\delta}(a_0)\big)$.*

Итак, вероятность переобучения слоя шара «сосредоточена на его внешней окружности». Построим два подмножества $S^m_{2\delta}(a_0)$, приближающих вероятность переобучения слоя шара. Обозначим через X^m множество объектов, на которых алгоритм a_0 допускает ошибку, $\bar{X}^m = \mathbb{X} \setminus X^m$.

Подмножество алгоритмов $B'(m, r_0)$ образуется всеми C^δ_m различными алгоритмами, которые допускают $m-\delta$ ошибок на X^m и δ ошибок на фиксированных объектах подвыборки $\bar{X}^m$. Очевидно, что оно целиком лежит в $S^m_{2\delta}(a_0)$.

Теорема 6. *Пусть μ — МЭР, $A = B'(m, r_0)$. Тогда для любого $\varepsilon \in [0, 1]$:*

$$Q_\varepsilon = \sum_{\substack{i=0\\ i+j+p=\ell}}^{m} \sum_{j=0}^{h} \sum_{p=0}^{\delta} \frac{C^i_m C^j_h C^p_\delta}{C^\ell_L} \times$$

$$\times \Big(\big[i < \delta\big]\big[p \leqslant s_d(\varepsilon)\big] + \big[i \geqslant \delta\big]\big[i + p \leqslant \delta + s_d(\varepsilon)\big] \Big),$$

где $h = L - m - \delta$, $s_d(\varepsilon) = \frac{\ell}{L}(m - \varepsilon k)$.

Подмножество алгоритмов $B''(m, r_0)$**.** Пусть $L - m$ кратно δ. Семейство B'' — это все $C^\delta_m \frac{L-m}{\delta}$ алгоритмов, допускающих $m - \delta$ ошибок на X^m и δ ошибок на $\bar{X}^m$ при том, что для любого $x \in \bar{X}^m$ существует единственный $a \in B''$: $I(a, x) = 1$. Семейство B'' также является подмножеством $S^m_{2\delta}(a_0)$.

Теорема 7. *Пусть μ — МЭР и $\ell < \frac{L-m}{\delta}$. Тогда $Q_\varepsilon\big(B''(m, r_0)\big) = Q_\varepsilon\big(B^m_{r_0}(a_0)\big)$.*

Численный эксперимент

Вероятность переобучения Q_ε нетрудно оценить эмпирически методом Монте-Карло, если вместо доли всех разбиений выборки взять долю разбиений из заданного случайного подмножества разбиений. В данном эксперименте бралась тысяча случайных разбиений. Сравнивались точные оценки вероятности переобучения множеств $B^m_{r_0}(a_0)$ мощности 809 876 и $B'(m, r_0)$ мощности 45 с эмпирическими оценками $B''(m, r_0)$ мощности 4 275 и множества, состоящего из D случайных представителей $S^m_{2\delta}(a_0)$. Результаты представлены на рисунке 4.

В этом примере условие теоремы 7 не выполняется. Тем не менее семейство B'' достаточно хорошо приближает вероятность переобучения слоя шара. Семейство B' показало худший результат. Это объясняется тем, что алгоритмы из B' не различаются на подвыборке $\bar{X}^m$ и, в отличие от B'', заполняют множество $S^m_{2\delta}(a_0)$ неравномерно. Стоит также заметить, что Q_ε случайного подмножества $S^m_{2\delta}(a_0)$ достигает значения слоя шара при $D \approx 70$, что указывает на избыточность семейства B''.

Рис. 4. Оценки Q_ε для множеств $B^m_{r_0}(a_0)$, B', B'' и случайного подмножества D алгоритмов из $S^m_{2\delta}(a_0)$, при $l = k = 100$, $m = 10$, $r_0 = 4$, $\varepsilon = 0{,}05$.

Выводы

Получены точные формулы для вероятности переобучения модельных семейств алгоритмов — хэммингова шара радиуса r и некоторых его подмножеств. Показано, что на внешней окружности слоя хэммингова шара можно выбрать «разреженное» подсемейство из небольшого числа (порядка десятков) алгоритмов, вероятность переобучения которого будет очень близка к вероятности переобучения всего слоя, состоящего из огромного числа алгоритмов. Таким образом, дано косвенное объяснение экспериментов [4], в которых измеренные значения эффективного локального коэффициента разнообразия также не превосходили нескольких десятков.

Есть основания полагать, что аппроксимация «плотных» семейств их «разреженными» подсемействами является перспективным подходом, который позволит оценивать вероятность переобучения в практических ситуациях. Данная работа является первым шагом в этом направлении.

Литература

[1] *Вапник В. Н., Червоненкис А. Я.* Теория распознавания образов. — М.: Наука, 1974.

[2] *Boucheron S., Bousquet O., Lugosi G.* Theory of classification: A survey of some recent advances // ESAIM: Prob. and Stat. — 2005. — No. 9. — Pp. 323–375.

[3] *Frey A. I.* Accurate estimates of the generalization ability for symmetric sets of predictors and randomized learning algorithms // Pattern Recognition and Image Analysis. — 2010. — Vol. 20, No. 3. — Pp. 241–250.

[4] *Vorontsov K. V.* Combinatorial probability and the tightness of generalization bounds // Patt. Rec. and Image An. — 2008. — Vol. 18, No. 2. — Pp. 243–259.

[5] *Vorontsov K. V.* Splitting and similarity phenomena in the sets of classifiers and their effect on the probability of overfitting // Pattern Recognition and Image Analysis. — 2009. — Vol. 19, No. 3. — Pp. 412–420.

[6] *Vorontsov K. V.* Exact combinatorial bounds on the probability of overfitting for empirical risk minimization // Pattern Recognition and Image Analysis. — 2010. — Vol. 20, No. 3. — Pp. 269–285.

Вероятность переобучения плотных и разреженных многомерных сеток алгоритмов*

Фрей А. И.
sashafrey@gmail.com
Московский Физико-технический институт

Известно, что для получения высокоточных оценок обобщающей способности в задачах обучения по прецедентам необходимо вводить более «тонкие» характеристики семейства алгоритмов, чем размерность. Предлагаются две такие характеристики — высота и разреженность. Показывается, что для монотонных и унимодальных многомерных сеток алгоритмов этих характеристик достаточно для получения точной оценки вероятности переобучения. Исследуется зависимость вероятности переобучения от размерности, высоты и разреженности для метода рандомизированной минимизации эмпирического риска.

The probability of overfitting for dense and sparse multidimensional grids of classifiers*

Frei A. I.
Moscow Institute of Physics and Technology, Moscow, Russia

The dimensional characteristics of the hypotheses set are known to be insufficient for obtaining tight generalization bounds. We propose two novel characteristics — the height and the sparsity of the hypotheses set, and show that they are sufficient to obtain exact generalization bounds for two special sets — the monotonic and unimodal multidimensional grids of classifiers. Then we study how the probability of overfitting depends on dimension, height, and sparsity in the case of randomized empirical risk minimization.

Введение

Для построения надёжных методов обучения по прецедентам необходимо иметь как можно более точные оценки обобщающей способности, например, в виде верхних оценок вероятности переобучения. Известные оценки используют различные характеристики семейства алгоритмов, метода обучения и обучающей выборки [4], в первую очередь, сложность (размерность) семейства алгоритмов. Точность таких оценок может быть недостаточной для практических применений [6]. В [7, 8] показано, что надёжное обучение возможно только для семейств, обладающих одновременно двумя свойствами — расслоением алгоритмов по частотам ошибок и сходством алгоритмов.

Эффект расслоения состоит в том, что при фиксированной обучающей выборке малую частоту ошибок имеет лишь малая доля алгоритмов. Вероятность получить алгоритм в результате обучения резко падает с ростом его частоты ошибок. Поэтому достаточно оценивать сложность лишь нижних слоёв семейства, иначе оценки вероятности переобучения окажутся сильно завышенными.

Эффект сходства состоит в том, что схожие алгоритмы вносят существенно меньший вклад в вероятность переобучения, чем несхожие. Поэтому вероятность переобучения у семейств, непрерывных по параметрам, может оказаться вполне приемлемой даже при высокой размерности.

В [7, 8] было также показано, что учёт этих двух эффектов по отдельности в общем случае не даёт численно точных оценок вероятности переобучения. Но при этом не было предложено каких-либо удобных (желательно, скалярных) численных характеристик расслоения и сходства. Векторную характеристику *профиля расслоения* пока удавалось оценивать только с помощью весьма трудоёмкого метода Монте-Карло [2].

В данной работе вводятся скалярные характеристики *высоты* и *разреженности* семейства алгоритмов, связанные со свойствами расслоения и сходства, соответственно. Рассматриваются модельные семейства — многомерные монотонные и унимодальные сетки алгоритмов, для которых этих двух характеристик (в дополнение к длине выборки и размерности) оказывается достаточно для получения точной оценки вероятности переобучения. Эти семейства впервые были рассмотрены в [1], а их одномерные частные случаи — монотонные и унимодальные цепочки алгоритмов — в [7, 8]. Отличие данной работы в том, что вводится понятие разреженности семейства и применяется теоретико-групповой подход [3] для рандомизированного (а не пессимистичного, как в [1]) метода минимизации эмпирического риска.

Таким образом, показана принципиальная возможность получения точных оценок вероятности переобучения, в которых свойства размерности, расслоения и сходства выражаются тремя скалярными характеристиками, соответственно.

Работа поддержана РФФИ (проект №08-07-00422) и программой ОМН РАН «Алгебраические и комбинаторные методы математической кибернетики и информационные системы нового поколения».

Постановка задачи

Пусть $\mathbb{X} = (x_i)_{i=1}^L$ — генеральная выборка, состоящая из L объектов. Отображения $a\colon \mathbb{X} \to \{0,1\}$ будем называть алгоритмами и говорить, что алгоритм a допускает ошибку на объекте x_i, если $a(x_i) = 1$. Каждому алгоритму взаимно однозначно соответствует бинарный вектор ошибок $\big(a(x_i)\big)_{i=1}^L$.

Величина $n(a,U) = \sum\limits_{x\in U} a(x)$ называется *числом ошибок* алгоритма a на подвыборке $U \subseteq \mathbb{X}$.

Величина $\nu(a,U) = n(a,U)/|U|$ называется *частотой ошибок* алгоритма a на подвыборке $U \subseteq \mathbb{X}$.

Обозначим через $\mathbb{A} = \{0,1\}^L$ множество всех 2^L бинарных векторов длины L, тогда $2^{\mathbb{A}}$ — это множество всех подмножеств $\mathbb{A}$.

Обозначим через $[\mathbb{X}]^\ell$ множество всех ℓ-элементных подмножеств генеральной выборки $\mathbb{X}$.

Детерминированным методом обучения назовем произвольное отображение $\mu\colon 2^{\mathbb{A}} \times [\mathbb{X}]^\ell \to \mathbb{A}$, которое по обучающей выборке $X \in [\mathbb{X}]^\ell$ выбирает из подмножества $A \subseteq \mathbb{A}$ некоторый алгоритм $a = \mu(A,X)$. Метод μ называется *минимизацией эмпирического риска* (МЭР), если выбираемый им алгоритм допускает наименьшее число ошибок на обучении: для всех $X \in [\mathbb{X}]^\ell$ и $A \subseteq \mathbb{A}$

$$\mu(A,X) \in A(X) \equiv \operatorname{Arg}\min_{a\in A} n(a,X).$$

Вопрос о том, какой именно алгоритм a из $A(X)$ выдать в результате обучения, может решаться по-разному. В пессимистичном методе МЭР выбирается алгоритм с максимальным $n(a,\mathbb{X})$, что приводит к верхним оценкам вероятности переобучения [8, 1]. Мы рассматриваем *рандомизированный метод обучения* [3, 5], который произвольным $A \in 2^{\mathbb{A}}$ и $X \in [\mathbb{X}]^\ell$ ставит в соответствие не один алгоритм a, а нормированную функцию $f(a)$, значение которой можно интерпретировать как вероятность получить алгоритм a в результате обучения:

$$\mu\colon 2^{\mathbb{A}} \times [\mathbb{X}]^\ell \to \Big\{ f\colon \mathbb{A} \to [0,1] \;\Big|\; \sum_{a\in\mathbb{A}} f(a) = 1 \Big\}. \quad (1)$$

Примером рандомизированного метода обучения является *рандомизированный метод МЭР*, основанный на равновероятном выборе a из $A(X)$:

$$\mu(A,X)(a) = \big[a \in A(X)\big] / \big|A(X)\big|. \quad (2)$$

Здесь и далее квадратные скобки переводят логическое выражение в число: [истина] = 1, [ложь] = 0.

Пусть $X \sqcup \bar{X} = \mathbb{X}$ — произвольное разбиение генеральной выборки на обучающую выборку $X \in [\mathbb{X}]^\ell$ и контрольную $\bar{X} = \mathbb{X}\backslash X$. *Уклонением частот* назовем разность частот ошибок на контроле и на обучении: $\delta(a,X) = \nu(a,\bar{X}) - \nu(a,X)$.

Зафиксируем параметр $\varepsilon \in (0,1]$.

Будем говорить, что алгоритм a *переобучен* при разбиении $X \sqcup \bar{X}$, если $\delta(a,X) \geqslant \varepsilon$.

Примем единственное вероятностное предположение, что все разбиения генеральной выборки $\mathbb{X}$ на две подвыборки — наблюдаемую обучающую X и скрытую контрольную $\bar{X}$ — равновероятны [8]. Тогда *вероятность переобучения* для детерминированного метода обучения μ определяется как

$$Q_\mu(\varepsilon, A) = \frac{1}{C_L^\ell} \sum_{X\in[\mathbb{X}]^\ell} \big[\delta(\mu(A,X),X) \geqslant \varepsilon\big],$$

а для рандомизированного метода μ — как

$$Q_\mu(\varepsilon, A) = \frac{1}{C_L^\ell} \sum_{X\in[\mathbb{X}]^\ell} \sum_{a\in A} \mu(A,X)(a) \big[\delta(a,X) \geqslant \varepsilon\big].$$

Монотонная сетка алгоритмов

Пусть $\boldsymbol{d} = (d_1,\ldots,d_h) \in \mathbb{Z}^h$ — целочисленный вектор индексов. Обозначим $\|\boldsymbol{d}\| = \max\limits_{j=1,\ldots,h} |d_j|$, $|\boldsymbol{d}| = |d_1| + \ldots + |d_h|$. На множестве векторов индексов введём покомпонентное отношение сравнения: $\boldsymbol{d} \leqslant \boldsymbol{d}'$, если $d_j \leqslant d'_j$, $j = 1,\ldots,h$; и $\boldsymbol{d} < \boldsymbol{d}'$ если хотя бы одно из неравенств $d_j \leqslant d'_j$ строгое.

Определение 1. *Множество алгоритмов* $A_M = \big\{a_{\boldsymbol{d}}\colon \boldsymbol{d} \geqslant \boldsymbol{0},\ \|\boldsymbol{d}\| \leqslant D\big\}$ *называется монотонной* h*-мерной сеткой алгоритмов высоты* D*, если* $\mathbb{X}$ *разбивается на непересекающиеся подмножества* U_0, U_1 *и* $X_j = \{x_j^1,\ldots,x_j^D\}$, $j = 1,\ldots,h$, *такие, что:*

1) $a_{\boldsymbol{d}}(x_j^i) = [i \leqslant d_j]$, *где* $x_j^i \in X_j$;
2) $a_{\boldsymbol{d}}(x_0) = 0$ *при всех* $x_0 \in U_0$;
3) $a_{\boldsymbol{d}}(x_1) = 1$ *при всех* $x_1 \in U_1$.

Монотонная сетка — это модель семейства алгоритмов с h непрерывными параметрами, предполагающая, что по мере непрерывного увеличения j-го параметра при фиксированных остальных ошибки возникают последовательно на объектах $x_j^1,\ldots,x_j^D$.

Обозначим $m \equiv |U_1|$. Из определения 1 следует, что $n(a_{\boldsymbol{d}},\mathbb{X}) = m + |\boldsymbol{d}|$. Число алгоритмов в h-мерной монотонной сетке высоты D составляет $(D+1)^h$. Алгоритм $a_{\boldsymbol{0}}$ является *лучшим в сетке*.

Пример 1. Двумерная ($h = 2$) монотонная сетка при $m = 0$ и $L = 4$:

	$a_{0,0}$	$a_{1,0}$	$a_{2,0}$	$a_{0,1}$	$a_{1,1}$	$a_{2,1}$	$a_{0,2}$	$a_{1,2}$	$a_{2,2}$
x_1	0	**1**	**1**	0	**1**	**1**	0	**1**	**1**
x_2	0	0	**1**	0	0	**1**	0	0	**1**
x_3	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**
x_4	0	0	0	0	0	0	**1**	**1**	**1**

Определение 2. *Разреженностью* ρ *множества алгоритмов* A *назовём минимальное хэммингово расстояние между векторами ошибок:*

$$\rho = \min_{a,a'\in A} \sum_{i=1}^L \big|a(x_i) - a'(x_i)\big|.$$

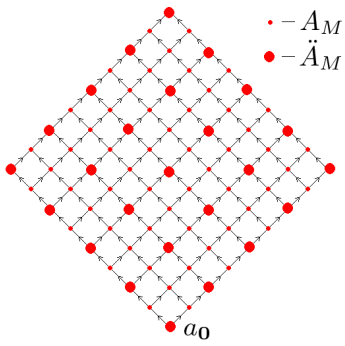

Рис. 1. Разреженная двумерная монотонная сетка. Узлы сетки соответствуют алгоритмам, направление стрелок — возрастанию числа ошибок алгоритмов.

Определение 3. *Разреженной h-мерной монотонной сеткой с разреженностью ρ называется подмножество* $\ddot{A}_M = \{a_{\boldsymbol{d}} \in A_M : \boldsymbol{d} \in (\rho\mathbb{Z})^h\}$.

Если исходная сетка A_M имела высоту D, то величину $\lfloor D/\rho \rfloor$ будем называть высотой сетки $\ddot{A}_M$.

Пример 2. На рис. 1 выделено подмножество алгоритмов двумерной монотонной сетки высоты $D = 8$, соответствующее разреженной монотонной сетке с параметрами $\rho = 2$, $D = 4$.

Введем дополнительные обозначения:

C_n^k — биномиальные коэффициенты, причём будем считать, что $C_n^k = 0$ при $k < 0$ и $k > n$;

$H_L^{\ell,m}(z) = \frac{1}{C_L^\ell} \sum\limits_{s=0}^{\lfloor z \rfloor} C_m^s C_{L-m}^{\ell-s}$ — функция гипергеометрического распределения, $z \in [0, \ell]$;

$Y_*^{h,D} \subset \mathbb{Z}^h$ — множество целочисленных невозрастающих неотрицательных последовательностей длины h, первый член которых не превосходит D;

S_h — группа перестановок элементов последовательности $\boldsymbol{\lambda} \in Y_*^{h,D}$;

$|S_h\boldsymbol{\lambda}|$ — мощность орбиты действия S_h на $\boldsymbol{\lambda}$.

Теорема 1. *Пусть $\ddot{A}_M$ — разреженная h-мерная монотонная сетка высоты D и разреженности ρ. Тогда вероятность переобучения $Q_\mu(\varepsilon, \ddot{A}_M)$ для рандомизированного метода минимизации эмпирического риска дается формулой*

$$Q_\mu(\varepsilon, \ddot{A}_M) = \sum_{\boldsymbol{\lambda} \in Y_*^{h,D}} \sum_{\substack{\boldsymbol{t} \geqslant \rho\boldsymbol{\lambda}, \\ \|\boldsymbol{t}\| \leqslant \rho D}} \frac{|S_h\boldsymbol{\lambda}|}{T(\lfloor \boldsymbol{t}/\rho \rfloor)} \frac{C_{L'}^{\ell'}}{C_L^\ell} H_{L'}^{\ell',m}(s_0),$$

где $\ell' = \ell - \sum\limits_{j=1}^{h} [t_j \neq \rho D]$, $k' = k - |\boldsymbol{t}|$, $L' = \ell' + k'$, $T(\boldsymbol{t}) = \prod\limits_j (t_j + 1)$, $s_0 = \frac{\ell}{L}(m + \rho|\boldsymbol{\lambda}| - \varepsilon k)$.

При $\rho = 1$ данная формула дает вероятность переобучения неразреженной сетки алгоритмов A_M.

Унимодальная сетка алгоритмов

Унимодальная сетка является более реалистичной моделью семейства с h непрерывными параметрами, по сравнению с монотонной сеткой. Предполагается, что непрерывное отклонение j-го параметра не только в бóльшую, но и в меньшую, сторону от оптимального значения приводит к увеличению числа ошибок.

Определение 4. *Множество алгоритмов* $A_U = \{a_{\boldsymbol{d}} : \|\boldsymbol{d}\| \leqslant D\}$ *называется унимодальной h-мерной сеткой алгоритмов высоты D, если $\mathbb{X}$ разбивается на непересекающиеся подмножества U_1, U_0, $X_j = \{x_j^1, \ldots, x_j^D\}$, $Y_j = \{y_j^1, \ldots, y_j^D\}$, $j = 1, \ldots, h$, такие, что:*

1) $a_{\boldsymbol{d}}(x_j^i) = [d_j > 0][i \leqslant |d_j|]$, *где* $x_j^i \in X_j$;
2) $a_{\boldsymbol{d}}(y_j^i) = [d_j < 0][i \leqslant |d_j|]$, *где* $y_j^i \in Y_j$;
3) $a_{\boldsymbol{d}}(x_0) = 0$ *при всех* $x_0 \in U_0$;
4) $a_{\boldsymbol{d}}(x_1) = 1$ *при всех* $x_1 \in U_1$.

Данное определение отличается от определения монотонной сетки отсутствием ограничения $\boldsymbol{d} \geqslant \mathbf{0}$. Число алгоритмов в h-мерной унимодальной сетке высоты D равно $(2D+1)^h$. Как и для монотонной сетки, число ошибок $n(a_{\boldsymbol{d}}, \mathbb{X})$ алгоритма $a_{\boldsymbol{d}}$ равно $m + |d|$, где $m \equiv |U_1|$.

Определение 5. *Разреженной h-мерной унимодальной сеткой с разреженностью ρ называется подмножество* $\ddot{A}_U = \{a_{\boldsymbol{d}} \in A_U : \boldsymbol{d} \in (\rho\mathbb{Z})^h\}$.

Если исходная сетка A_U имела высоту D, то величину $\lfloor D/\rho \rfloor$ будем называть высотой сетки $\ddot{A}_U$.

Обозначим через $n(\boldsymbol{\lambda})$ число ненулевых компонент последовательности $\boldsymbol{\lambda} \in Y_*^{h,D}$.

Теорема 2. *Пусть $\ddot{A}_U$ — разреженная h-мерная унимодальная сетка высоты D и разреженности ρ. Тогда вероятность переобучения $Q_\mu(\varepsilon, \ddot{A}_U)$ для рандомизированного метода минимизации эмпирического риска дается формулой*

$$Q_\mu(\varepsilon, \ddot{A}_U) = \sum_{\boldsymbol{\lambda} \in Y_*^{h,D}} \sum_{\substack{\boldsymbol{t} \geqslant \rho\boldsymbol{\lambda}, \\ \|\boldsymbol{t}\| \leqslant \rho D}} \sum_{\substack{\boldsymbol{t}' \geqslant \mathbf{0}, \\ \|\boldsymbol{t}'\| \leqslant \rho D}} \mathbb{S}(\boldsymbol{\lambda}, \boldsymbol{t}, \boldsymbol{t}'),$$

$$\mathbb{S}(\boldsymbol{\lambda}, \boldsymbol{t}, \boldsymbol{t}') = \frac{|S_h\boldsymbol{\lambda}| \cdot 2^{n(\boldsymbol{\lambda})}}{T(\lfloor \boldsymbol{t}/\rho \rfloor + \lfloor \boldsymbol{t}'/\rho \rfloor)} \frac{C_{L'}^{\ell'}}{C_L^\ell} H_{L'}^{\ell',m}(s_0),$$

$\ell' = \ell - \sum\limits_{j=1}^{h} \big([t_j \neq \rho D] + [t'_j \neq \rho D]\big)$, $k' = k - |\boldsymbol{t}| - |\boldsymbol{t}'|$, *остальные обозначения те же, что в теореме 1.*

Вычислительный эксперимент

На рис. 2 показана зависимость вероятности переобучения h-мерной монотонной сетки от ее высоты. При увеличении высоты сетки свыше $D = 5$ вероятность переобучения выходит на константу. Поэтому дальнейшие графики построены при малом значении параметра $D = 3$.

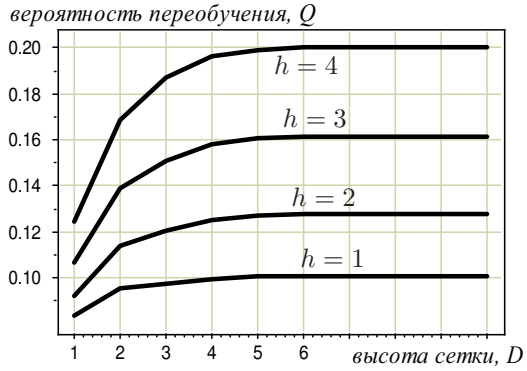

Рис. 2. Зависимость $Q_\mu(\varepsilon, \ddot{A}_M)$ от высоты D монотонной сетки при $L = 150,\ \ell = 90,\ \varepsilon = 0.05,\ m = 5,\ \rho = 1$.

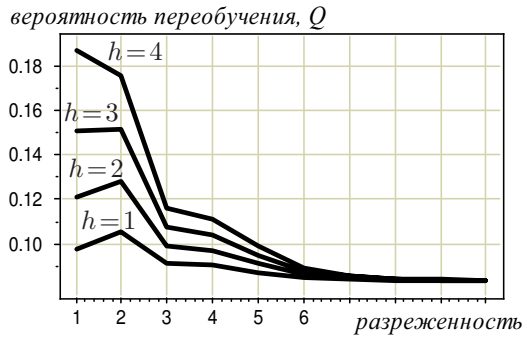

Рис. 3. Зависимость $Q_\mu(\varepsilon, \ddot{A}_M)$ от разреженности ρ при $L = 150,\ \ell = 90,\ \varepsilon = 0{,}05,\ D = 3,\ m = 5$.

На рис. 3 изображена зависимость вероятности переобучения h-мерной монотонной сетки при $h = 1, 2, 3, 4$ от разреженности ρ. При увеличении разреженности ρ вероятность переобучения падает, и вскоре выходит на константу, соответствующую вероятности переобучения лучшего алгоритма семейства a_0. Это связано с тем, что с увеличением ρ вероятность получить в результате обучения алгоритм a_0 стремится к единице.

На рис. 4 приведены результаты сравнения разреженных h-мерных унимодальных сеток с разреженными $2h$-мерными монотонными сетками, при $h = 1$ и $h = 2$. Серая кривая соответствует вероятности переобучения унимодальной сетки. Результаты подтверждают гипотезу [1] о том, что вероятность переобучения унимодальной сетки и монотонной сетки двойной размерности очень близки.

Выводы

В работе предложены два новых параметра множества алгоритмов — *разреженность* и *высота*. Для разреженных монотонных и унимодальных сеток показано, что эти параметры, наряду с длиной выборки и размерностью, определяют вероятность переобучения. Получены точные формулы вероятности переобучения для плотных ($\rho = 1$) и разреженных ($\rho > 1$) многомерных монотонных и унимодальных сеток в случае рандомизированной минимизации эмпирического риска. С помощью полученных формул экспериментально установлено, что в широком диапазоне параметров для монотонных и унимодальных сеток вероятность переобучения определяется несколькими нижними слоями семейства. Также показано, что с увеличением разреженности семейства вероятность получить лучший алгоритм в результате обучения стремится к единице.

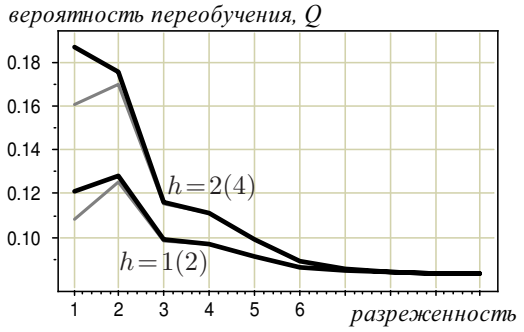

Рис. 4. Сравнение $Q_\mu(\varepsilon, \ddot{A}_M)$ и $Q_\mu(\varepsilon, \ddot{A}_U)$ от ρ при $L = 150,\ \ell = 90,\ \varepsilon = 0{,}05,\ D = 3,\ m = 5,\ h = 1(2), 2(4)$.

Литература

[1] *Ботов П. В.* Точные оценки вероятности переобучения для монотонных и унимодальных семейств алгоритмов // Всеросс. конф. ММРО-14. — М.: МАКС Пресс, 2009. — С. 7–10.

[2] *Кочедыков Д. А.* Структуры сходства в семействах алгоритмов классификации и оценки обобщающей способности // Всеросс. конф. ММРО-14. — М.: МАКС Пресс, 2009. — С. 45–48.

[3] *Фрей А. И.* Точные оценки вероятности переобучения для симметричных семейств алгоритмов // Всеросс. конф. ММРО-14. — М.: МАКС Пресс, 2009. — С. 66–69.

[4] *Boucheron S., Bousquet O., Lugosi G.* Theory of classification: A survey of some recent advances // ESAIM: Probability and Statistics. — 2005. — No. 9. — Pp. 323–375.

[5] *Frey A. I.* Accurate Estimates of the Generalization Ability for Symmetric Sets of Predictors and Randomized Learning Algorithms // Pattern Recognition and Image Analysis. — 2010. — Vol. 20, No. 3. — Pp. 241–250.

[6] *Vorontsov K. V.* Combinatorial probability and the tightness of generalization bounds // Pattern Recognition and Image Analysis. — 2008. — Vol. 18, No. 2. — Pp. 243–259.

[7] *Vorontsov K. V.* Splitting and similarity phenomena in the sets of classifiers and their effect on the probability of overfitting // Pattern Recognition and Image Analysis. — 2009. — Vol. 19, No. 3. — Pp. 412–420.

[8] *Vorontsov K. V.* Exact combinatorial bounds on the probability of overfitting for empirical risk minimization // Pattern Recognition and Image Analysis. — 2010. — Vol. 20, No. 3. — Pp. 269–285.

Вероятность переобучения конъюнкций пороговых предикатов над вещественными признаками*

Ивахненко А. А.
andrej_iv@mail.ru
Москва, Вычислительный центр им. А. А. Дородницына РАН

В данной работе рассматривается задача классификации с помощью взвешенного голосования предикатов из семейства пороговых условий. Описывается структура семейства этих предикатов. Для данного семейства предикатов выводится верхняя оценка функционала качества предиката. Показывается эффективный способ вычисления оценки.

Overfitting probability of threshold predicate conjunctions on real feature values*

Ivahnenko A. A.
Institution of Russian Academy of Sciences Dorodnicyn Computing Centre of RAS, Moscow, Russia

В данной работе рассматриваются алгоритмы классификации, основанные на поиске логических закономерностей по эмпирическим данным.

Целью работы является установление зависимости между свойствами алгоритмов на обучающей и контрольных выборках.

При решении задач классификации с неточными, неполными, разнородными данными одной из основных проблем является обеспечение наилучшего возможного качества классификации вне обучающей выборки.

Задача классификации

В данной работе задача классификации ставится следующим образом.
$\mathbb{X}$ — множество *объектов*;
$Y = \{0, 1\}$ — множество возможных ответов;
$X^L = \{x_i\},\ i = 1, \ldots, L,\ x_i \in \mathbb{X}$ — *полная выборка*;
$Y^L = \{y_i\},\ i = 1, \ldots, L,\ y_i \in Y$ — *вектор ответов*.

Признаком называется отображение $f\colon \mathbb{X} \to D_f$, описывающее результат измерения некоторой характеристики объекта, где D_f — заданное множество. В данной работе рассматриваются признаки, для которых $D_f = \mathbb{R}$.

Пусть имеется набор признаков $f_1, \ldots, f_n$. Вектор $(f_1(x), \ldots, f_n(x))$ называют признаковым описанием объекта $x \in \mathbb{X}$. В дальнейшем мы не будем различать объекты из $\mathbb{X}$ и их признаковые описания, полагая $\mathbb{X} = D_{f_1} \times \ldots \times D_{f_n}$. Совокупность признаковых описаний всех объектов выборки X^L, записанную в виде таблицы размером $L \times n$, называют *матрицей объектов–признаков*: $F = ||f_j(x_i)||,\ i = 1, \ldots, L,\ j = 1, \ldots, n$.

Допустим, что значения каждого признака f_j на всех объектах выборки X^L попарно различны. Пусть $f_j(x_j^{(i)})$ – i-ый элемент в вариационном ряду значений j-ого признака $f_j(x_j^{(1)}) < \cdots < f_j(x_j^{(L)})$. Построим по матрице F целочисленную матрицу F' заменив каждое значение $f_j(x_j^{(i)})$ на порядковый номер i. В матрице F' сохранен порядок объектов по каждому признаку и она удобнее для дальнейшего рассмотрения, т.к. номер значения по порядку совпадает с самим значением. В дальнейшем будем использовать матрицу F' в качестве матрицы объектов-признаков.

Алгоритмом классификации называется функция $\varphi\colon \mathbb{X}, W \to Y$, где W – множество параметров. В данной работе в качестве алгоритма классификации используется взвешенное голосование закономерностей $\varphi(x) = \arg\max\limits_{y \in Y} \sum_{a \in R_y} w_a a(x)$, где R_y – множество предикатов a класса y, $w_a \in \mathbb{R}$ – вес голоса предиката a. Говорят что предикат $a \in R_y$ относит объект $x \in X$ к классу $y \in Y$, если $a(x) = 1$. Будем рассматривать предикаты из семейства:

$$A = \Big\{ a(x; c_1, \ldots, c_n) = \bigwedge_{j=1}^{n} [x^j \leqslant_j c_j] \colon$$
$$x = (x^1, \ldots, x^n) \in X^L,\ c_j \in \mathbb{R},\ \leqslant_j \in \{\leqslant, \geqslant\} \Big\},$$

где $x^j \equiv f_j(x)$ — значение j-го признака объекта x. Каждому предикату a соответствует L-мерный бинарный вектор ошибок $\bar{a} = \big(a(x_i) \neq y_i\big)_{i=1}^{L}$. Имеет смысл рассматривать только предикаты, для которых пороги c_j берутся из множества значений j-ого признака $(1, \ldots, L)$ и нуля, поэтому число различных векторов ошибок для данной матрицы X^L не превысит $(L+1)^n$. Введем отношение эквивалентности на множестве предикатов $a \sim a' \leftrightarrow \bar{a} = \bar{a}'$. Структура классов эквивалентности будет рассмотрена дальше. Множество векторов ошибок будем обозначать так:

$$\bar{A} = \{\bar{a},\ a \in A\}.$$

Процесс подбора предикатов и настройки их порогов по обучающей выборке $X^\ell \subset X^L$ называ-

Работа выполнена при финансовой поддержке РФФИ, проект № 00-00-00000.

ют *настройкой* или *обучением. Методом обучения предиката* называется отображение $\mu\colon (X \times Y)^\ell \to a$, которое произвольной конечной выборке X^ℓ ставит в соответствие предикат $a\colon X \to Y$. Будем говорить также, что метод μ строит предикат a по выборке X^ℓ и записывать $a = \mu(X)$ или $a = \mu X$.

Как правило, обучение сводится к поиску параметров модели, доставляющих оптимальное значение заданному функционалу качества.

Число ошибок предиката a на произвольной выборке $X \subseteq X^L$ есть $n(a, X) = \sum_{x_i \in X} [a(x_i) \neq y_i]$.

Частота ошибок предиката a на произвольной выборке $X \subseteq X^L$ есть $\nu(a, X) = \frac{n(a,X)}{|X|}$.

Если предикат a доставляет минимум функционалу $\nu(a, X^\ell)$ на заданной обучающей выборке $X^\ell \subset X^L$, то это ещё не гарантирует, что он будет иметь малую частоту ошибок на контрольной выборке $X^k = X^L \backslash X^\ell$. Для построения хорошего алгоритма $\varphi(x)$ нужно уметь подбирать предикаты имеющие малую частоту ошибок на контроле. Поиск способа отбора таких предикатов и является главной целью исследования в данной работе.

Когда частота ошибок на контроле $\nu(a, X^k)$ оказывается существенно больше частоты ошибок на обучении $\nu(a, X^\ell)$, говорят об эффекте *переобучения* или переподгонки. Также говорят, что данный предикат обладает плохой *обобщающей способностью.*

Разобъём полную выборку $X^L = \{x_i\}$, $i = 1, \ldots, L$ всеми C_L^ℓ способами на две части: $X^L = X^\ell \sqcup X^k$ — обучающую подвыборку длины ℓ и контрольную длины k, где $\ell + k = L$. Договоримся, что $\sum\limits_{X^\ell, X^k}$ обозначает сумму по всем C_L^ℓ разбиениям. В данной работе для определения обобщающей способности будем использовать функционал характеризующий вероятность переобучения:

$$Q_\varepsilon(\mu, X^L) = \frac{1}{C_L^\ell} \sum_{X^\ell, X^k} \big[\nu(\mu X^\ell, X^k) - \nu(\mu X^\ell, X^\ell) > \varepsilon\big]. \tag{1}$$

Постановка задачи

Для решения задачи классификации обычно используется поиск оптимального в смысле некоторого функционала качества представителя параметрического семейства алгоритмов. При этом оптимизация использует как минимум два механизма: отбор признаков и оптимизации параметров модели алгоритма. Нашей целью является предложение методики совмещения этих механизмов так, чтобы информация о семействе алгоритмов, полученная в процессе оптимизации, использовалась при отборе признаков.

Структура классов эквивалентности семейства предикатов

Будем говорить что классы эквивалентности предикатов связаны, если векторы ошибок представителей этих классов различаются на одном объекте. Число классов эквивалентности предикатов, и связи между ними зависят от значений признаков всех объектов выборки x_i, и не зависят от их классификации $y_i, i = 1, \ldots, L$.

Пусть u и v – произвольные объекты, будем говорить что объект u *доминируется* объектом v по координате j, если $u^j < v^j$. Будем говорить что объект u *доминируется* множеством объектов $S \subseteq X^L$, если для каждого $j \in \{1, \ldots, n\}$ существует объект $s \in S$, такой, что $u^j < s^j$ (записывается: $u \prec S$). Если множество S состоит из одного элемента s и $u \prec S$, то будем писать $u \prec s$.

Определение 1. *Недоминирующимся подмножеством (НП) будем называть такое подмножество $S \subseteq X^L$, что любой объект $s \in S$ не доминируется подмножеством $S\backslash s$. Обозначим множество всех таких подмножеств мощности q через M_q,*

$$M_q = \left\{S \subseteq X^L \colon |S| = q,\ \forall s \in S\ s \not\prec S\backslash s\right\}.$$

Введем искусственный НП S_0, который будет состоять из одного объекта, $x_0 = (0, \ldots, 0)$. Обозначим $M_0 = \{S_0\}$. Предикат $a(x; 0, \ldots, 0)$ будем обозначать a_0.

Очевидно, мощность НП $|S|$ не может превышать n. Если S является НП, то любое подмножество $S' \subset S$ также является НП. Из этого утверждения следует эффективный способ построения всех множеств НП: для построения M_{q+1} добавим к каждому множеству из M_q один объект, еще не входящий в него; и если полученное множество является НП, то оно войдет в M_{q+1}. Также получается и грубая оценка числа множеств НП: $|M_0| = 1$, $|M_1| = L$, $|M_{q+1}| \leqslant \frac{L-q}{q+1}|M_q|$, $q = 1, \ldots, n-1$.

Лемма 1. *Для любого объекта x из НП S найдется хотя бы один признак $j \in \{1, \ldots, n\}$, по которому на данном x достигается $\max\limits_{s \in S} s^j$:*

$$\bigcup_{j=1}^{n} \operatorname{Arg}\max_{s \in S}(s^j) = S.$$

Поставим в соответствие подмножеству S предикат $a(x, S) = a(x; \max\limits_{x \in S} x^1, \ldots, \max\limits_{x \in S} x^n)$. Очевидно, разным S ставятся в соответствие разные $a(x, S)$, т. к. значения каждого признака x_i^j, $i = 1, \ldots, L$ попарно различны, следовательно, зная j и x_i^j, можно однозначно указать объект x_i. Следовательно, набор параметров $c_j = \max\limits_{x \in S} x^j$, $j = 1, \ldots, n$ задает подмножество S однозначно, и различным S не могут соответствовать одинаковые $a(x, S)$.

Пример 1. Рассмотрим задачу с $n = 2$ признаками и $L = 10$ объектами (рис. 1) и семейство предикатов

$$A = \{a(x; c_1, c_2) = [x^1 \leqslant c_1 \wedge x^2 \leqslant c_2] \colon x \in X^L\}.$$

Поскольку каждый из параметров c_1, c_2 принимает значения $0, \ldots, L$, $|A| = (L+1)^2$, и каждый предикат из A задается парой порогов (c_1, c_2). Таким образом, каждому предикату из семейства можно поставить в соответствие узел прямоугольной сетки $H = \{0, \ldots, L\}^2$. Объекты выборки также лежат в $X^L \subset H$, при этом каждый объект обладает уникальным значением каждого из своих признаков в силу того, что значения признаков попарно различны. Это означает, что никакие два объекта из X^L не могут лежать на одной вертикали или на одной горизонтали в сетке H, см. рис. 1.

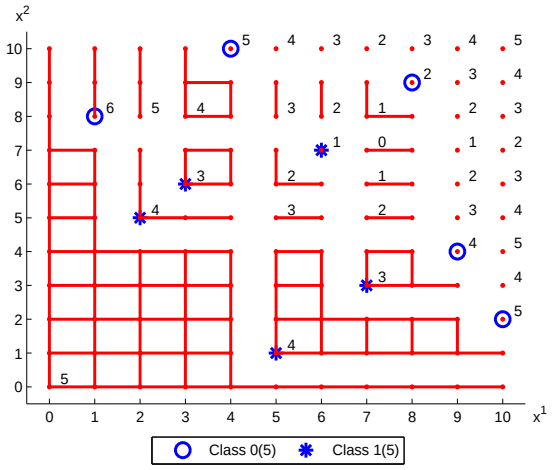

Рис. 1. Векторы ошибок предикатов обозначены точками, координаты которых совпадают с порогами предиката (c_1, c_2), соответствующего данному вектору. Одинаковые векторы ошибок соединены линиями и образуют классы эквивалентности. Около каждого класса эквивалентности подписано, сколько ошибок допускают предикаты этого класса на выборке X^L. На рисунке также отмечены объекты выборки.

Рассмотрим множество объектов на рис. 1. Подмножествами взаимно не доминирующих объектов мощности 1 являются все объекты выборки. Подмножествам взаимно не доминирующих объектов мощности 2 является, например, подмножество объектов $(2, 5)$ и $(5, 1)$. Объект $(5, 1)$ имеет большее значение первого признака, объект $(2, 5)$ – второго. Подмножество из объектов $(2, 5)$ и $(3, 6)$ не является НП, т. к. первый объект доминируется вторым. НП мощности больше 2 нет, т. к. размерность задачи $n = 2$.

Лемма 2. *Пусть $E \subset A$ — класс эквивалентности. Тогда предикат*

$$a_E(x) = a(x; \min_{a' \in E} c_1(a'), \ldots, \min_{a' \in E} c_n(a'))$$

принадлежит E, где $c_j(a')$ – j-ый параметр предиката a'

Будем называть предикат $a_E(x)$ *стандартным представителем* класса эквивалентности E. В частности, при размерности $n = 2$ на рис. 1 у каждого класса эквивалентности стандартным представителем является предикат $a(x; c_1, c_2)$, соответствующий левой нижней точке с координатами (c_1, c_2). Например $(5, 5); (7, 3); (0, 0)$.

Теорема 3. *Существует взаимно однозначное соответствие между множеством всех классов эквивалентности и множеством всех подмножеств недоминирующих объектов.*

Следствие 1. *Для каждого класса эквивалентности E существует и притом единственное НП S, такое что $a_E(x) = a(x, S)$*

Следствие 2. *Число классов эквивалентности предикатов равно $\sum_{q=0}^{n} |M_q|$.*

Оценка вероятности переобучения

Для оценки функционала $Q_\varepsilon(\mu, X^L)$ будем использовать методологию, описанную в [1]. В этой работе выдвигается гипотеза о том, что предикат a будет выбран на разбиении $X^L = X^\ell \sqcup X^k$ тогда и только тогда, когда объекты, называемые *порождающими*, принадлежат обучающей выборке X^ℓ, а объекты, называемые *запрещающими*, принадлежат контрольной выборке X^k:

$$[\mu X^\ell = a] = \sum_{v \in V_a} c_{av} [X_{av} \subseteq X^\ell] [X'_{av} \subseteq X^k], \qquad (2)$$

где X_{av} – множество порождающих объектов, X'_{av} – множество запрещающих объектов, V_a – множество индексов пар порождающих и запрещающих множеств для предиката a.

Рассмотрим множества пар порождающих и запрещающих множеств. Будем называть множества объектов $X_a = \bigcap_{v \in V_a} X_{av}$ и $X'_a = \bigcap_{v \in V_a} X'_{av}$ *фиксированными для предиката a*. Это объекты, которые необходимо должны присутствовать во всех порождающих или запрещающих множествах для того, чтобы метод обучения μ выбрал предикат a. Тогда гипотезу 2 можно переписать в виде оценки:

$$[\mu X^\ell = a] \leqslant [X_a \subseteq X^\ell] [X'_a \subseteq X^k]$$

Пусть для предиката a, множества X_a и X'_a являются фиксированными на обучении и контроле

соответственно. Число ошибок предиката a на объектах не из фиксированных множеств m_a. Число не фиксированных объектов на обучении $\ell_a = \ell - |X_a|$, число не фиксированных объектов на контроле $k_a = k - |X'_a|$, число не фиксированных объектов на всей выборке $L_a = \ell_a + k_a$. Наибольшее число ошибок предиката a на выборке без фиксированных объектов $s_a(\varepsilon) = \frac{\ell_a}{L_a}(n(a, X^L \backslash (X_a \cup X'_a)) - \varepsilon k_a)$. Доля разбиений для которых $X_a \subseteq X^\ell$ и $X'_a \subseteq X^k$ равна $P_a = C^{\ell_a}_{L_a}/C^\ell_L$, см. рис. 2. Справедлива следующая оценка:

Теорема 4.

$$Q_\varepsilon(\mu, X^L) \leqslant \sum_{a \in A} P_a H^{\ell_a, m_a}_{L_a}(s_a(\varepsilon)).$$

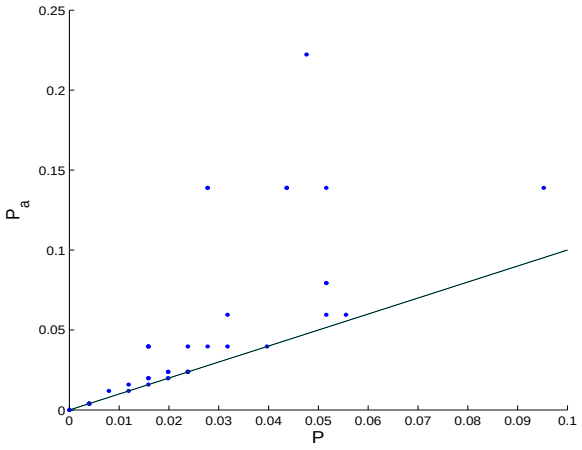

Рис. 2. На графике точками изображены предикаты. По оси абсцисс отложена вероятность выбора данного предиката, по оси ординат – оценка вероятности выбора по числу фиксированных точек P_a. Линия имеет уравнение $P = P_a$.

Лемма 5. *Объекты из НП S, такого, что $a(x; S) \sim a(x)$ являются фиксированными для предиката a. Причем объекты класса* 1 *принадлежат X_a, а объекты класса* 0 *– X'_a.*

В классической теории Вапника [2] есть оценка с такой же структурой $Q_\varepsilon \leqslant \sum_{a \in A} H^{\ell, m_a}_L(\frac{\ell}{L}(m_a - \varepsilon k))$. Оценка в теореме 4 более точна за счет учета структуры семейства предикатов. С другой стороны она отличается от точной формулы только вероятностью P_a. Чем точнее будет вычислена вероятность, тем точнее будет оценка. На рисунке 2 показно что предложенный вариант вычисления P_a является не сильно завышенным.

Выводы

В данной работе рассмотрена структура семейства предикатов пороговых условий. Выведена грубая оценка числа предикатов в семействе в зависимости от состава выборки. Предложен эффективный алгоритм перечисления всех классов эквивалентности предикатов. Выведена верхняя оценка функционала вероятности переобучения, которая может быть эффективно посчитана и лучше чем оценка Вапника за счет учета структуры семейства предикатов. Предложена постановка задачи использования структуры семейства предикатов для более точного решения задачи классификации. Дальнейшие исследования будут направлены на увеличение точности оценки и разработки методики применения результатов данной работы для решения задачи классификации.

Литература

[1] *Воронцов К. В.* Комбинаторный подход к проблеме переобучения.

[2] *Вапник В. Н.* Восстановление зависимостей по эмпирическим данным.— М.: Наука, 1979.